BỘ THÔNG TIN & TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN 2

BÁO CÁO LAB 3

IoT VÀ ỨNG DỤNG

DEMO NHẬN DIỆN ĐEO KHẨU TRANG

THẠC SĨ, GIẢNG VIÊN: ĐÀM MINH LỊNH

NHÓM 20

NGUYỄN DƯƠNG PHI – N20DCCN125 – D20CQCNPM02–N
NGUYỄN VŨ QUANG – N20DCCN130 – D20CQCNPM02–N

TP. HỒ CHÍ MINH, THÁNG 12 – 2023

– LỜI NÓI ĐẦU –

Trong bối cảnh đại dịch COVID–19 vẫn đang diễn biến phức tạp, việc đeo khẩu trang là một biện pháp quan trọng để bảo vệ bản thân và cộng đồng. Tuy nhiên, việc đeo khẩu trang không đúng cách có thể làm giảm hiệu quả phòng chống dịch. Do đó, việc phát triển hệ thống nhận diện đeo khẩu trang chính xác và hiệu quả là rất cần thiết.

Hiện nay, có nhiều phương pháp để nhận diện đeo khẩu trang, chẳng hạn như sử dụng camera hồng ngoại, camera truyền thống hoặc cảm biến radar. Mỗi phương pháp đều có những ưu điểm và nhược điểm riêng.

Camera hồng ngoại có thể phát hiện người không đeo khẩu trang ngay cả trong điều kiện ánh sáng yếu, nhưng chi phí cao và khó lắp đặt. Camera truyền thống có chi phí thấp và dễ lắp đặt, nhưng hiệu quả nhận diện kém trong điều kiện ánh sáng yếu. Cảm biến radar có thể phát hiện người đeo khẩu trang không đúng cách ngay cả trong điều kiện ánh sáng yếu, nhưng chi phí cao và khó lắp đặt.

ESP32–CAM là một mạch microcontroller tích hợp camera, cho phép chúng ta dễ dàng xây dựng các ứng dụng nhận dạng hình ảnh. ESP32–CAM có giá thành hợp lý, dễ lắp đặt và sử dụng, và có hiệu quả nhận diện tốt trong điều kiện ánh sáng yếu.

Trong bài báo cáo này, chúng ta sẽ xem xét việc sử dụng ESP32–CAM để xây dựng một hệ thống nhận diện đeo khẩu trang. Hệ thống này sẽ sử dụng camera của ESP32–CAM để chụp ảnh khuôn mặt, phân tích ảnh để xác định vị trí của khẩu trang và xác định xem người đeo khẩu trang có đeo đúng cách hay không. Sau đó, hệ thống sẽ gửi kết quả nhận diện lên một máy chủ hoặc nền tảng IoT để xử lý và lưu trữ.

– LỜI CẢM ƠN –

– Chúng em xin chân thành cảm ơn thầy Đàm Minh Lịnh trong thời gian qua đã dạy, hướng dẫn giúp đỡ chúng em trong quá trình học tập. Giúp chúng em nắm vững kiến thức của môn IoT VÀ ỨNG DỤNG. Từ đó chúng em có cái nhìn sâu hơn về lập trình ứng dụng IoT và công nghệ thông tin.

– Chúc thầy luôn luôn mạnh khỏe, luôn vui tươi, dồi dào sức sống và có nhiều thành công trong công việc giảng dạy!

– Cuối cùng nhóm 20 chúng em xin cảm ơn gia đình, người thân và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình học tập và thực hiện bài lab này.

– MỤC LỤC –

I. ĐỀ TÀI: LAB DEMO NHẬN DIỆN ĐEO KHẨU TRANG (Face Mask Detection) 5
II. CHUẨN BỊ: 5
1. PHẦN CỨNG: 5
2. PHẦN MỀM: Arduino IDE (v1.8.18) 5
3. DỊCH VỤ: 5
III. TÌM HIỂU: 5
1. ĐẶC TẢ HỆ THỐNG – SƠ ĐỒ: 5
2. TEACHABLE MACHINE: 5
IV. QUY TRÌNH THỰC HIỆN: 7
1. NỐI MẠCH: 7
2. CÀI ĐẶT ARDUINO IDE: 7
a. KHÁI NIỆM: 7
b. CÀI ĐẶT: 7
c. KHỞI CHẠY & THIẾT LẬP: 9
3. CÀI ĐẶT TELEGRAM: 13
a. KHÁI NIỆM: 13
b. CÀI ĐẶT: 14
4. TRAINING MODEL TEACHABLE MACHINE: 16
5. CODE: 20
a. facemask.ino: 20
b. index.h: 29
V. KẾT QUẢ: 37

I. <u>ĐỀ TÀI</u>: LAB DEMO NHẬN DIỆN ĐEO KHẨU TRANG (Face Mask Detection)

II. <u>CHUẨN BỊ</u>:

1. <u>PHẦN CỨNG</u>:

– Mạch nạp ESP32–CAM.

– Mạch thu phát Wifi BLE ESP32–CAM.

– Module Camera OV2640.

– Cáp Micro USB.

2. <u>PHẦN MỀM</u>: Arduino IDE (v1.8.18)

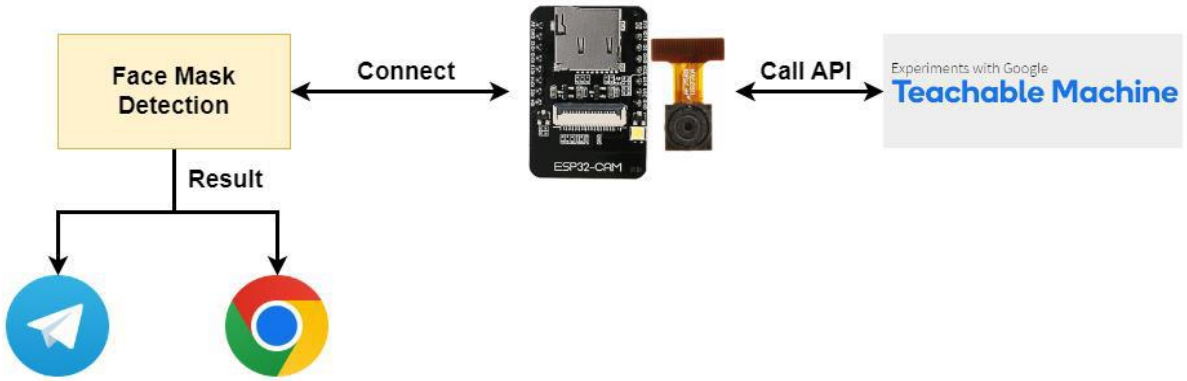
3. <u>DỊCH VỤ</u>:

– Teachable Machine.

– Telegram.

III. <u>TÌM HIỂU</u>:

1. <u>ĐẶC TẢ HỆ THỐNG – SƠ ĐỒ</u>:



Mô hình hoạt động của hệ thống Face Mask Detection bao gồm các thành phần như bên trên:

– Dùng thiết bị ESP32–CAM kết nối với máy tính.

– Dùng Arduino nạp Code cho thiết bị ESP32–CAM.

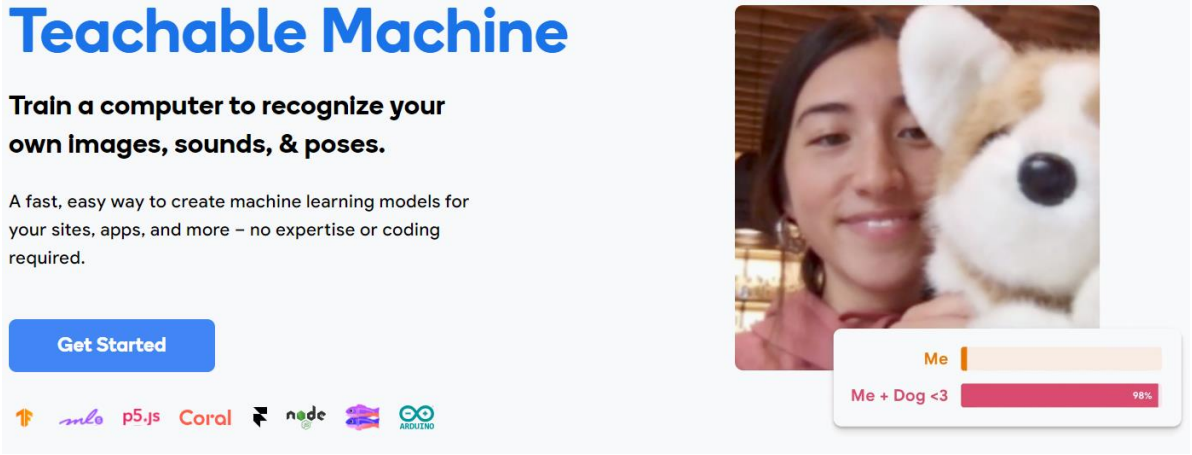
– ESP32–CAM sử dụng API của Techable Machine.

– Thực hiện chức năng phát hiện mang khẩu trang và trả kết quả về Telegram.

2. <u>TEACHABLE MACHINE</u>:

– Teachable Machine là một công cụ trực tuyến được phát triển bởi Google AI, cho phép người dùng tạo ra các mô hình học máy cơ bản. Công cụ này sử dụng giao diện kéo và thả đơn giản, giúp người dùng có thể dễ dàng tạo và đào tạo các mô hình phân loại ảnh, nhận dạng đối tượng và dự đoán giá trị.

– Theo Google, Teachable Machine được "phát triển với thư viện deeplearn.js giúp các nhà phát triển web tìm hiểu về máy học một cách dễ dàng bằng cách huấn luyện và chạy các mạng nơ–ron trực tiếp trong trình duyệt".



– Teachable Machine bao gồm các tính năng sau:
  + **Đào tạo mô hình học máy từ đầu:** Người dùng có thể tạo và đào tạo các mô hình học máy từ đầu bằng cách cung cấp dữ liệu đào tạo cho công cụ. Dữ liệu đào tạo có thể là hình ảnh, âm thanh hoặc video.
  + **Sử dụng các mô hình có sẵn:** Người dùng cũng có thể sử dụng các mô hình có sẵn từ thư viện của Teachable Machine
  + **Xem trước mô hình:** Người dùng có thể xem trước kết quả của mô hình trước khi triển khai. Điều này giúp người dùng có thể đảm bảo rằng mô hình hoạt động chính xác như mong đợi.
  + **Tải xuống mô hình:** Người dùng có thể tải xuống mô hình để sử dụng trong các ứng dụng khác.

– Teachable Machine được sử dụng cho nhiều ứng dụng khác nhau, bao gồm:
  + **Phân loại ảnh:** Teachable Machine có thể được sử dụng để phân loại ảnh thành các danh mục khác nhau, chẳng hạn như động vật, thực vật hoặc đồ vật.
  + **Nhận dạng đối tượng:** Teachable Machine có thể được sử dụng để nhận dạng các đối tượng trong ảnh, chẳng hạn như khuôn mặt, biển báo hoặc sản phẩm.
  + **Dự đoán giá trị:** Teachable Machine có thể được sử dụng để dự đoán giá trị của một số thứ, chẳng hạn như giá cả của sản phẩm hoặc điểm số của học sinh.

– Teachable Machine là một công cụ học máy tuyệt vời cho những người mới bắt đầu. Nó giúp người dùng có thể dễ dàng tạo và đào tạo các mô hình học máy cơ bản mà không cần có kiến thức về học máy. Với sự phổ biến của công nghệ AI và học máy, việc sử dụng Teachable Machine cũng mở ra cơ hội cho việc áp dụng học máy trong nhiều lĩnh vực khác nhau như y tế, giáo dục, và công nghiệp.

IV. QUY TRÌNH THỰC HIỆN:

1. NỐI MẠCH:

– Lắp mạch thu phát Wifi BLE ESP32–CAM vào mạch nạp.

– Lắp module camera OV2640 vào khe FPC Connector trên mạch thu phát.

Hình IV.1.1. Mạch nạp và mạch thu phát.

Hình IV.1.2. Lắp mạch hoàn chỉnh.

2. CÀI ĐẶT ARDUINO IDE:

a. KHÁI NIỆM:

– Arduino IDE là một phần mềm mã nguồn mở chủ yếu được sử dụng để viết & biên dịch mã vào module Arduino. Giúp cho việc biên dịch mã dễ dàng mà ngay cả một người bình thường không có kiến thức kỹ thuật cũng có thể làm được.

– Nó có các phiên bản cho các hệ điều hành như MAC, Windows, Linux và chạy trên nền tảng Java đi kèm với các chức năng và lệnh có sẵn đóng vai trò quan trọng để gỡ lỗi, chỉnh sửa và biên dịch mã trong môi trường.

– Có rất nhiều các module Arduino như Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro và nhiều module khác.

– Mỗi module chứa một bộ vi điều khiển trên bo mạch được lập trình và chấp nhận thông tin dưới dạng mã.

– Mã chính, còn được gọi là sketch, được tạo trên nền tảng IDE sẽ tạo ra một file Hex, sau đó được chuyển và tải lên trong bộ điều khiển trên bo.

– Môi trường IDE chủ yếu chứa hai phần cơ bản: Trình chỉnh sửa và Trình biên dịch, phần đầu sử dụng để viết mã được yêu cầu và phần sau được sử dụng để biên dịch và tải mã lên module Arduino. Môi trường này hỗ trợ cả ngôn ngữ C và C ++.

– Khi người dùng viết mã và biên dịch, IDE sẽ tạo file Hex cho mã. File Hex là các file thập phân Hexa được Arduino hiểu và sau đó được gửi đến bo mạch bằng cáp USB. Mỗi bo Arduino đều được tích hợp một bộ vi điều khiển, bộ vi điều khiển sẽ nhận file hex và chạy theo mã được viết. [1]

b. CÀI ĐẶT:

– Tải phần mềm ở đây: https://www.arduino.cc/en/software

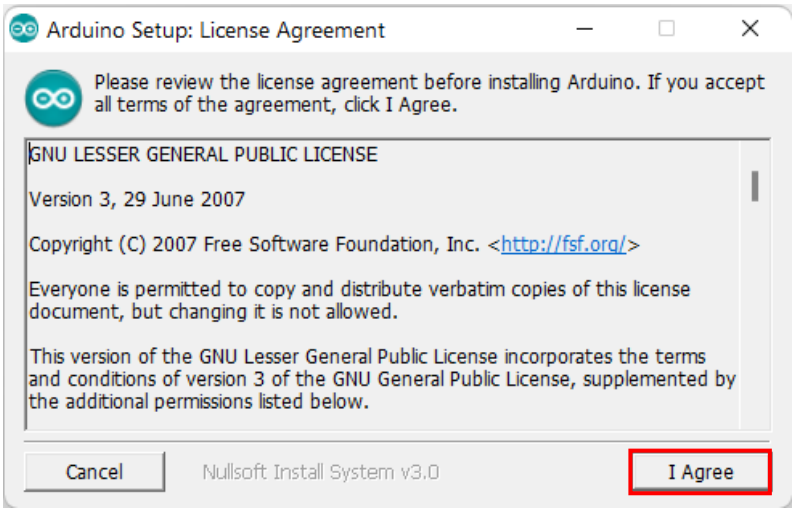
– (Bài LAB được thực hiện bằng Arduino IDE phiên bản 1.8.18. Tải tại đây: https://www.arduino.cc/en/software/OldSoftwareReleases)

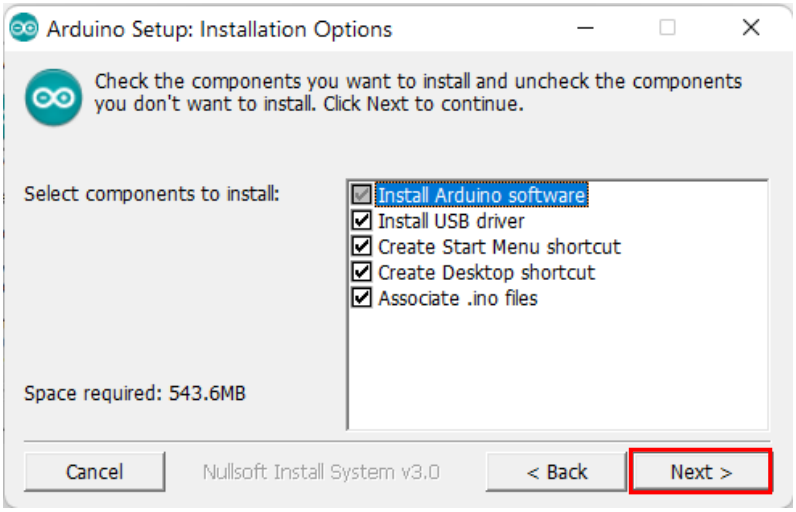
ARDUINO 1.8.18

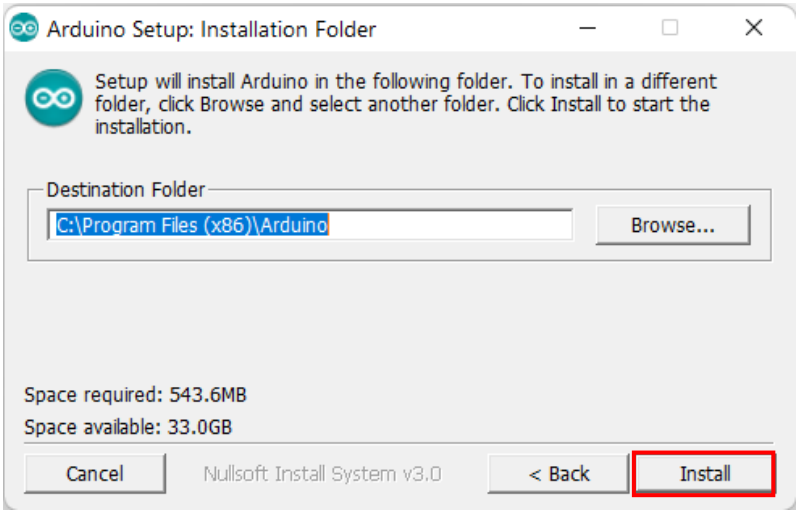
Arduino IDE that can be used with any Arduino board, including the Arduino Yún and Arduino DUE. Refer to the **Getting Started** page for Installation instructions. **See the release notes**.

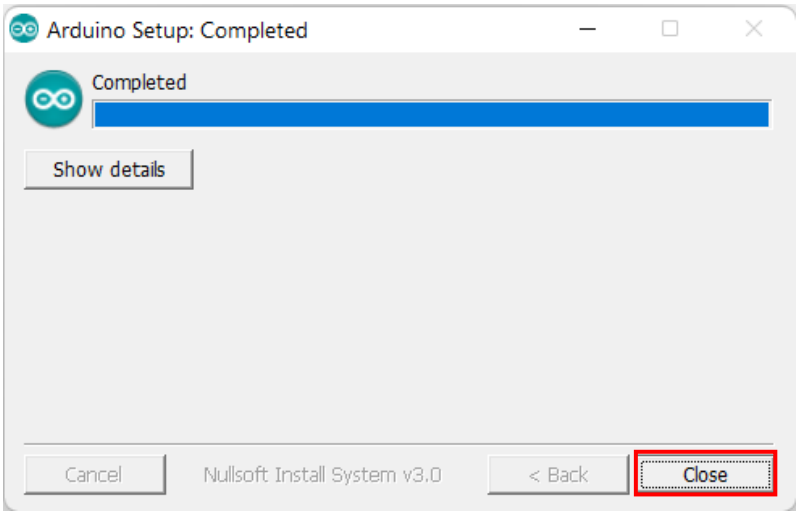
Windows	MAC	Linux	Source
Windows Installer **Windows ZIP file for non admin install**	**MAC OS 10.8 Mountain Lion or newer**	**Linux 32 bits** **Linux 64 bits** **Linux ARM 32** **Linux ARM 64**	**Source**

– Chạy File .exe để tiến hành cài đặt chương trình. Các bước cài đặt chương trình như hình dưới đây.



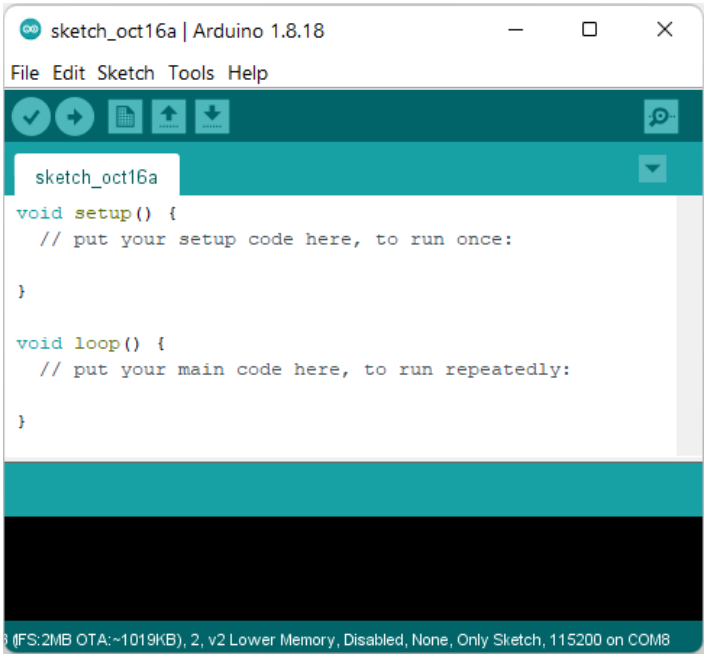






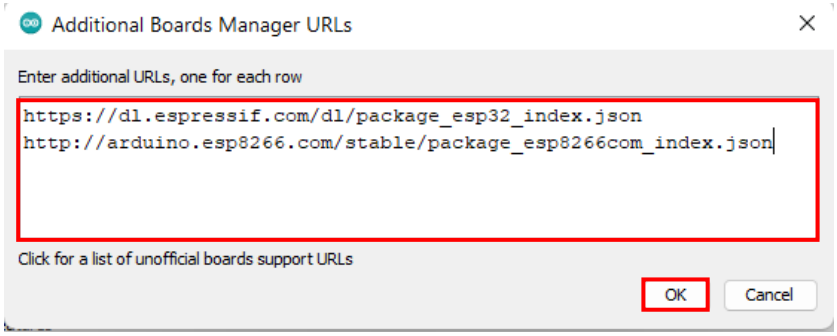
c. KHỞI CHẠY & THIẾT LẬP:

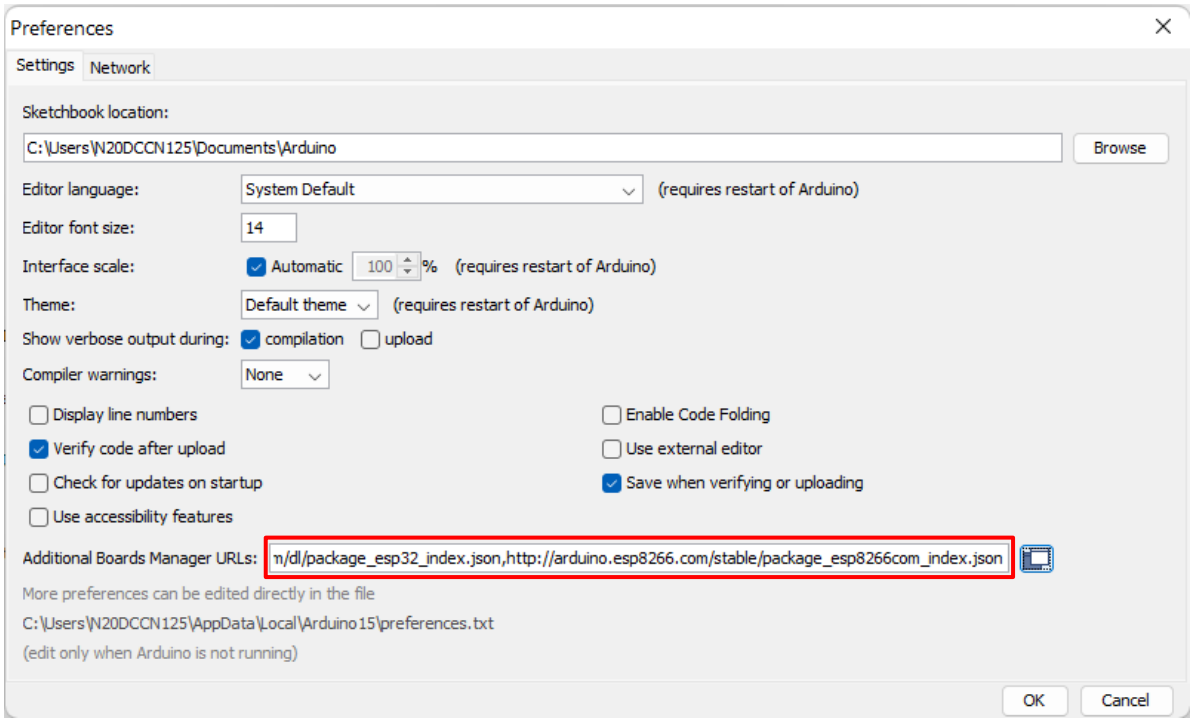
– Mở Arduino IDE lên, đây là giao diện chính.



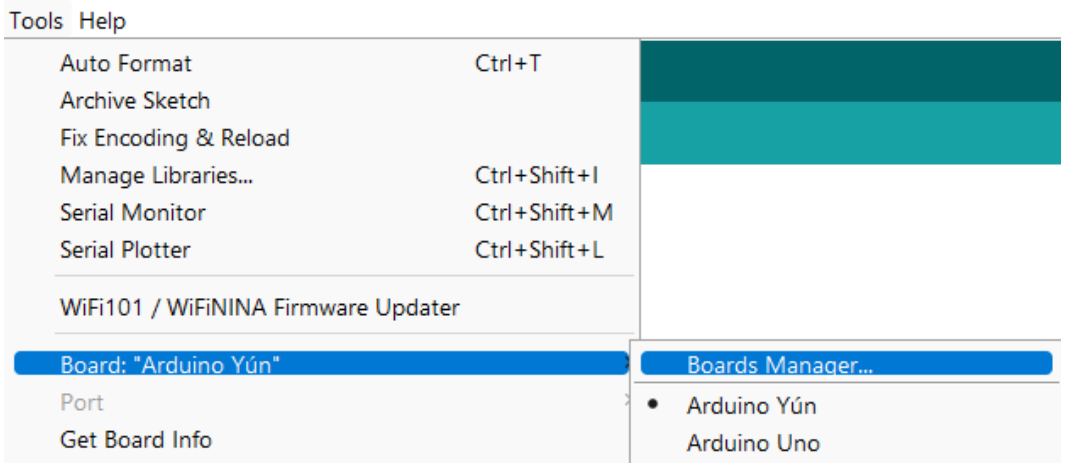
– Tiếp theo, ta cài đặt thư viện. Vào **File → Preferences**, điền 2 link sau vào ô **"Additional Boards Manager URLs"**:

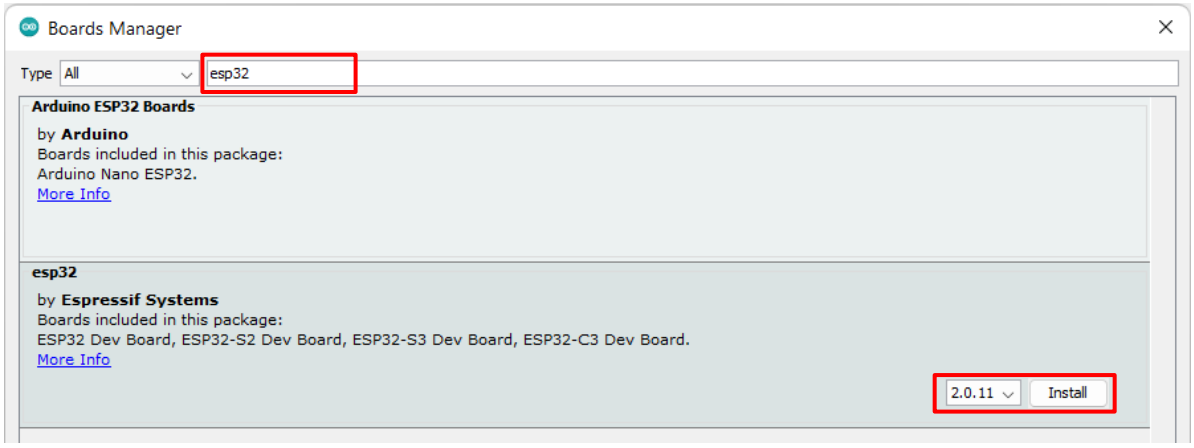
https://dl.espressif.com/dl/package_esp32_index.json
http://arduino.esp8266.com/stable/package_esp8266com_index.json



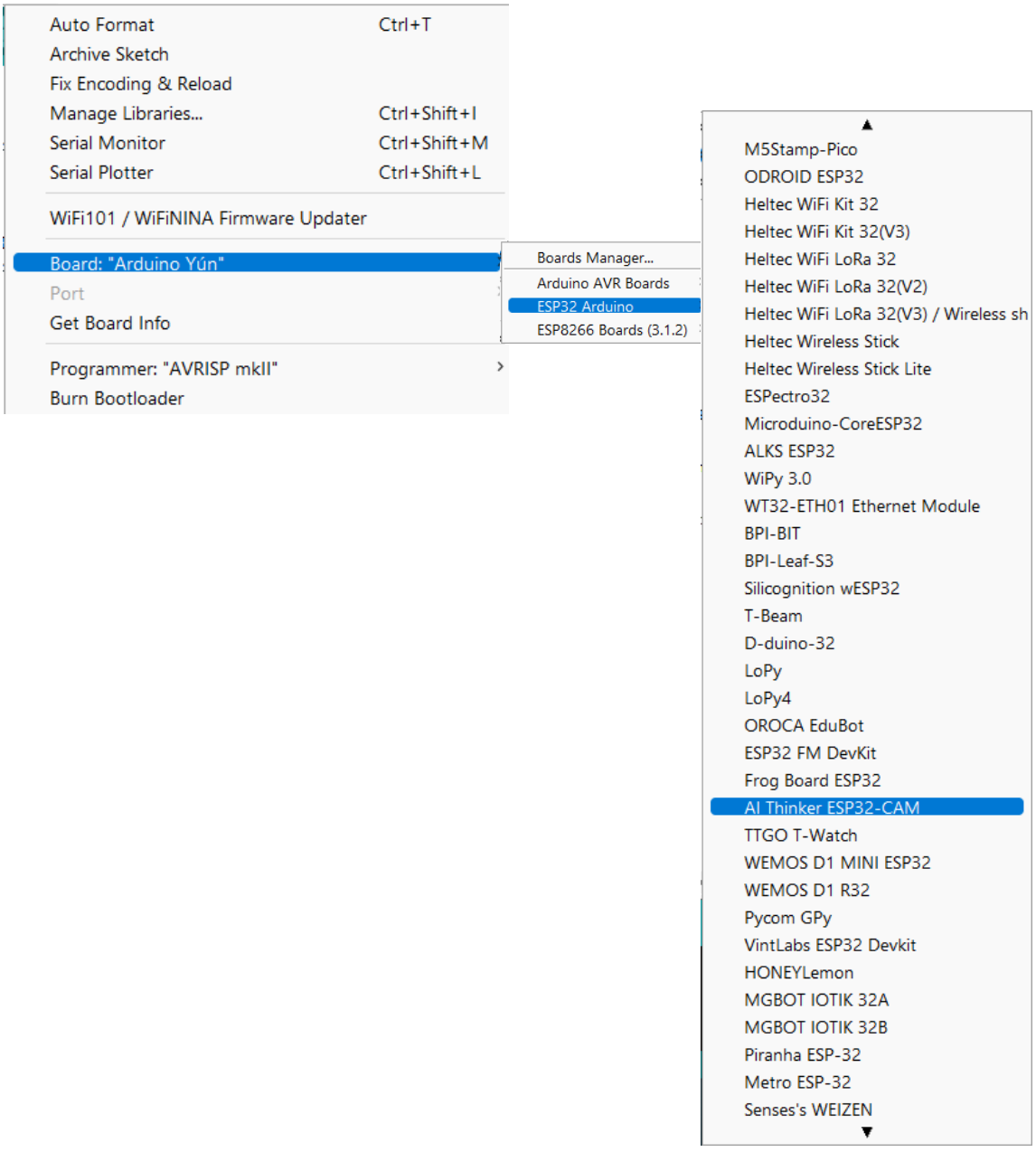


– Sau đó, ta vào **Tools → Board → Boards Manager.** Cài đặt thư viện ESP32.

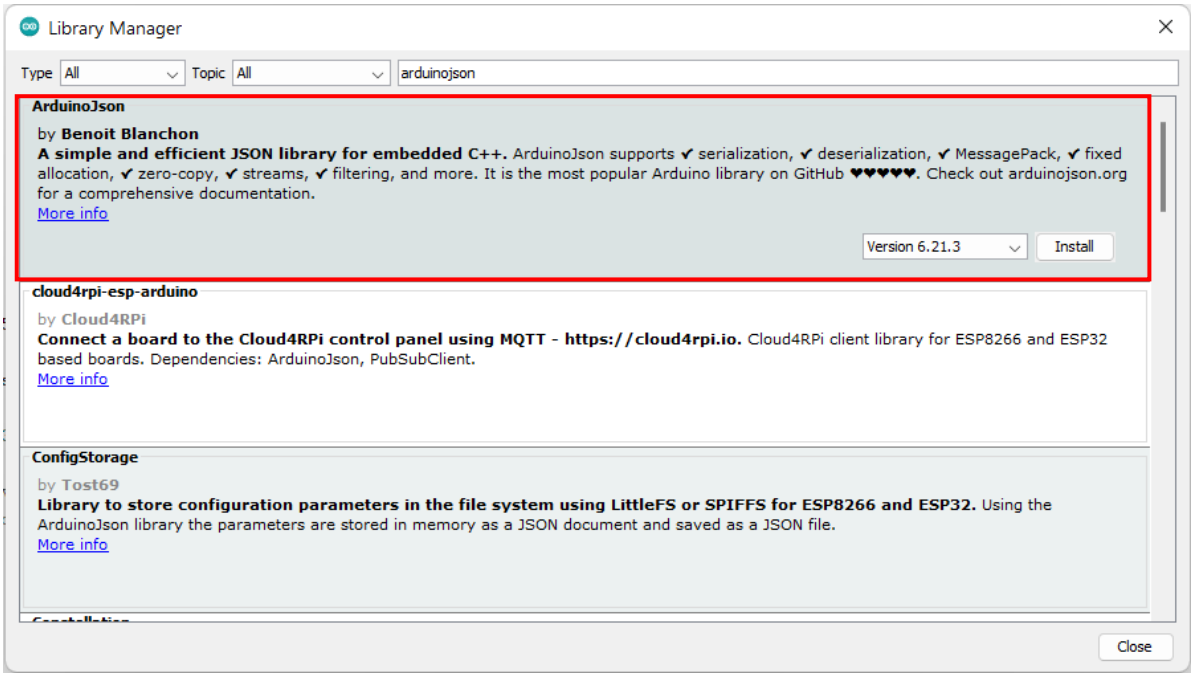




– Cài xong thư viện, ta chọn lại loại Board của Arduino bằng cách vào **Tools → Board → ESP32 Arduino → AI Thinker ESP32–CAM.**

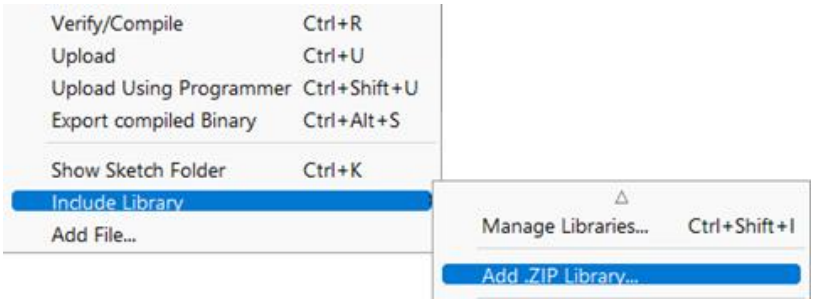


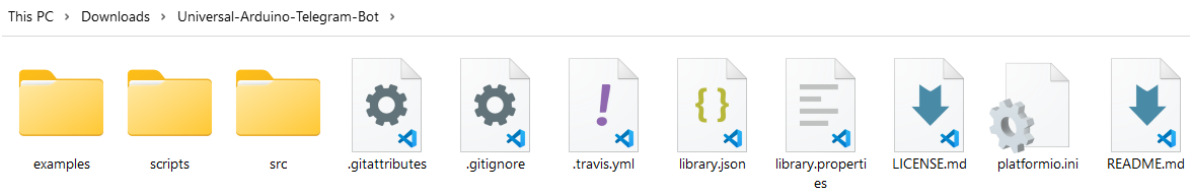
– Tiếp tục, ta tiến hành chọn **Sketch → Include Library → Manage Libraries,** tìm từ khóa "**ArduinoJson**" và tiến hành cài đặt. Trong bài nhận diện đeo khẩu trang sử dụng ESP–32CAM và bot Telegram, thư viện **ArduinoJson** được sử dụng để chuyển đổi dữ liệu từ camera ESP–32CAM sang định dạng JSON. Dữ liệu này sau đó được gửi đến bot Telegram để xử lý.



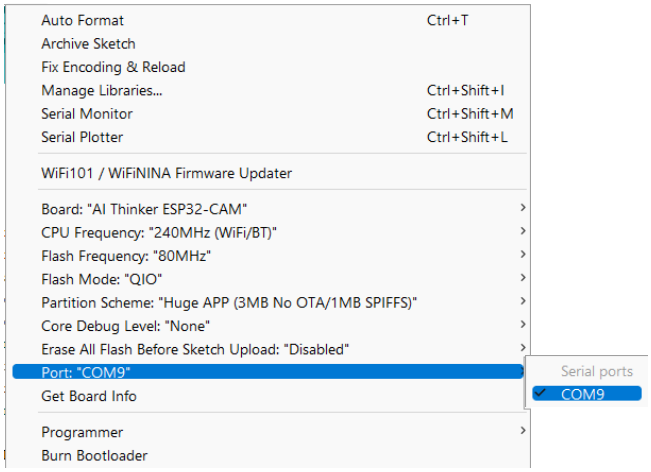
– Xong, ta import thư viện **Universal–Arduino–Telegram–Bot**, là một thư viện Arduino cho phép tạo và sử dụng các bot Telegram trên Arduino. Thư viện này cung cấp một giao diện đơn giản để tương tác với API Telegram Bot, giúp dễ dàng tạo các bot Telegram cho các ứng dụng IoT. Chọn **Sketch → Include Library → Add .ZIP Library…**

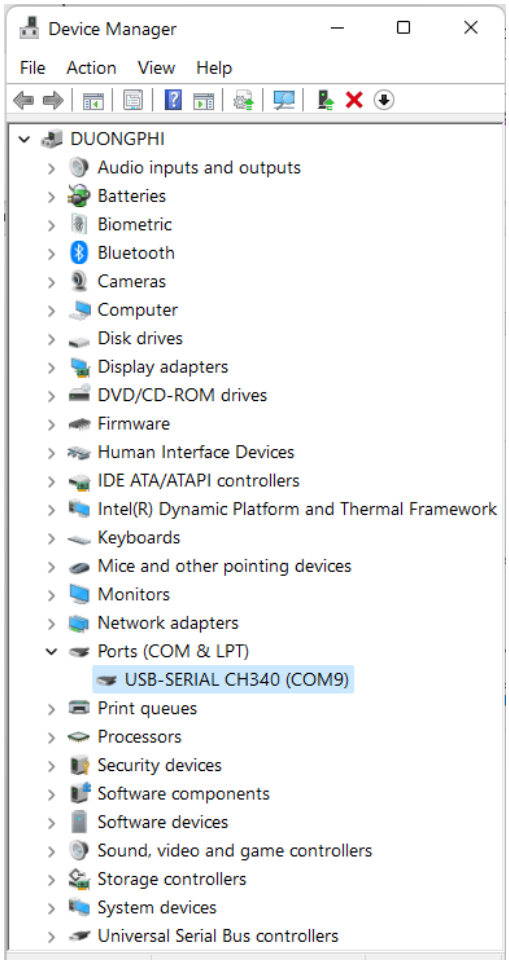
URL GitHub: https://github.com/witnessmenow/Universal–Arduino–Telegram–Bot





– Tiếp theo, ta tiến hành thiết lập cổng COM giữa Arduino và Laptop. Đảm bảo rằng ta đã cắm cáp vào cổng USB. Ở **Windows**, ta vào **Device Manager**, xuống tìm phần **Ports (COM & LPT)**, nếu xuất hiện như hình dưới là thành công **(USB–SERIAL CH340)**. Trong **Arduino IDE**, vào **Tool → Port**. Kiểm tra xem đã hiện PORT như hình hay chưa. (Ở đây Laptop nhận **COM9**, tùy theo máy tính)

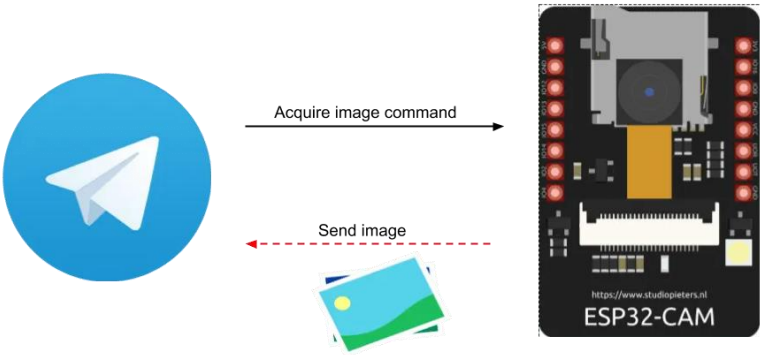




3. CÀI ĐẶT TELEGRAM:

a. KHÁI NIỆM:

– Telegram Messenger là một nền tảng nhắn tin đám mây cho phép người dùng gửi tin nhắn, gọi điện, chia sẻ tệp và hơn thế nữa. Telegram cũng cho phép người dùng tạo các bot, là các ứng dụng của bên thứ ba có thể được sử dụng để tự động hóa các tác vụ hoặc cung cấp thông tin.



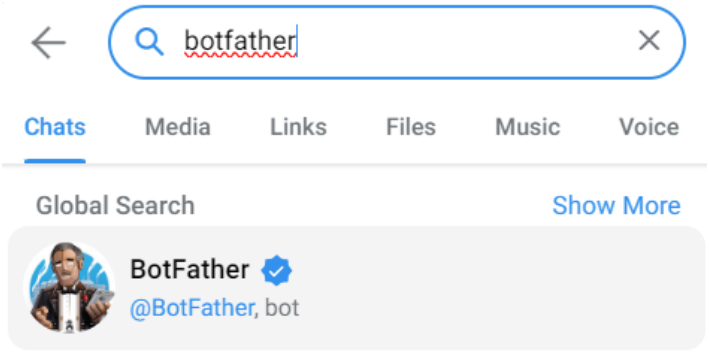
– ESP32–CAM là một vi điều khiển có tích hợp camera. Nó được sử dụng để tạo ra nhiều loại thiết bị, bao gồm camera giám sát, hệ thống báo động và hệ thống thông báo.

– Để tương tác với các bot Telegram, ESP32–CAM sử dụng API Telegram Bot, một API REST cho phép các nhà phát triển tương tác với các bot Telegram. API này cung cấp một số phương thức khác nhau để gửi và nhận tin nhắn, điều khiển đầu ra của ESP32–CAM, và truy cập dữ liệu từ cảm biến.

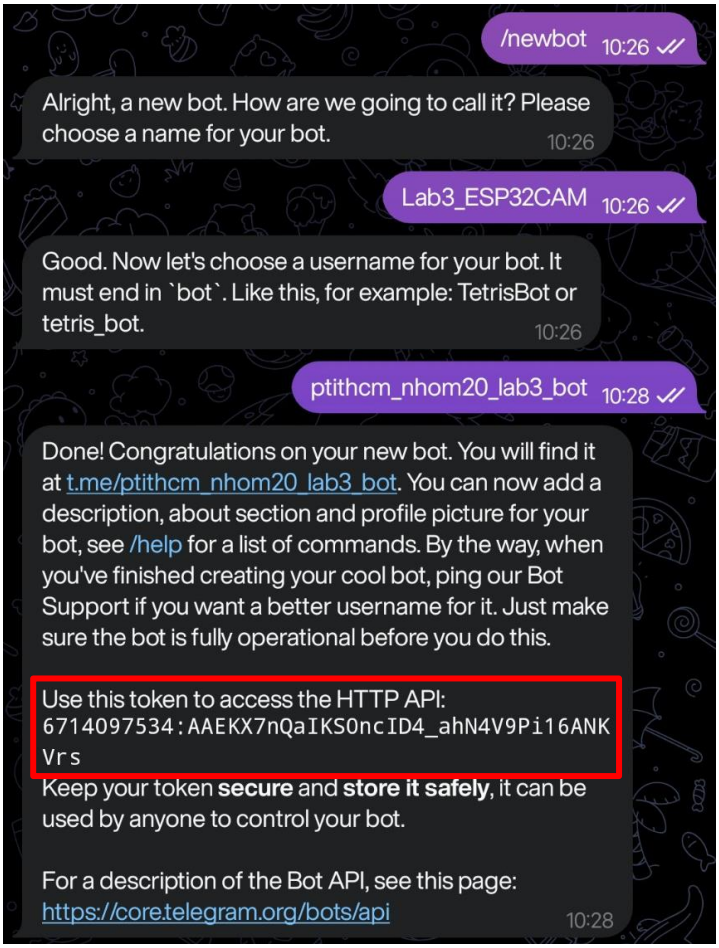
b. <u>CÀI ĐẶT</u>:

– Truy cập GooglePlay (Android) hoặc AppStore (iOS), tải xuống và cài đặt ứng dụng Telegram. Hoặc sử dụng Telegram Web trên máy tính: https://web.telegram.org/

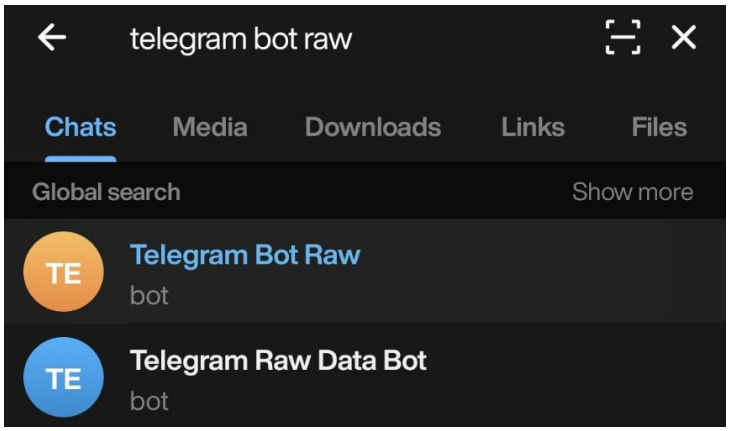
– Tiến hành đăng nhập/đăng ký tài khoản, sau đó tìm kiếm từ khóa **"BotFather",** và chọn vào bot như hình dưới.



– Nhập **/newbot** để tiến hành quá trình tạo bot mới. Sau đó làm theo hướng dẫn.
Sau khi nhập **/newbot**, đặt một tên cho bot, ví dụ: **"Lab3_ESP32CAM".**
Tiếp theo, đặt username cho bot, kết thúc username bằng từ "bot", ví dụ: **"ptithcm_nhom20_lab3_bot".**



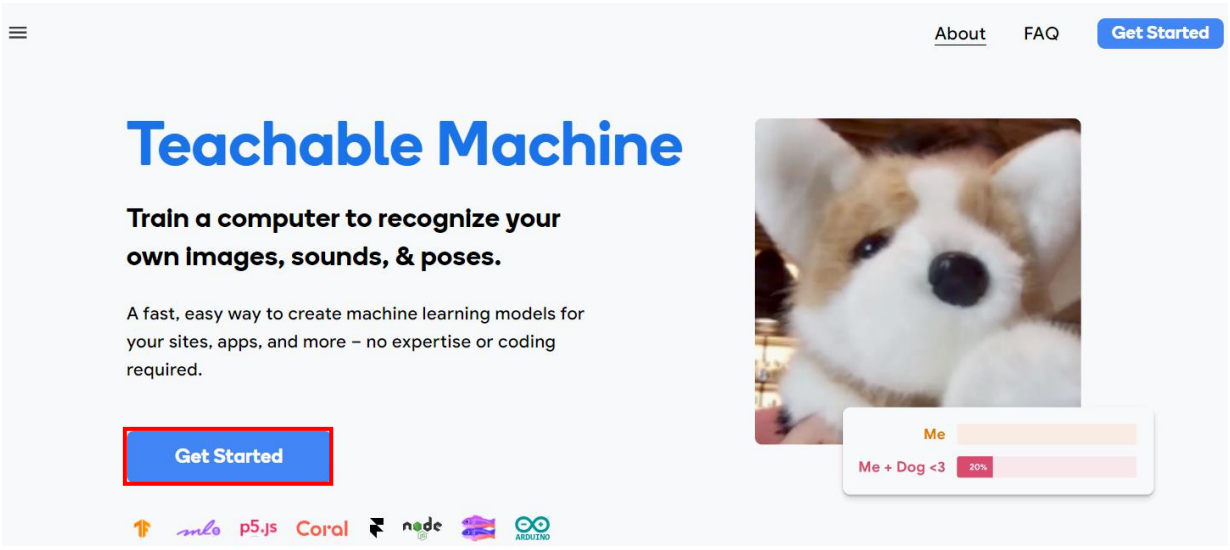
– Lưu ý mã token như trên hình.
– Bước tiếp theo, lấy ID của chat. Tìm kiếm từ khóa **"TelegramBotRaw",** và chọn vào bot như hình dưới.



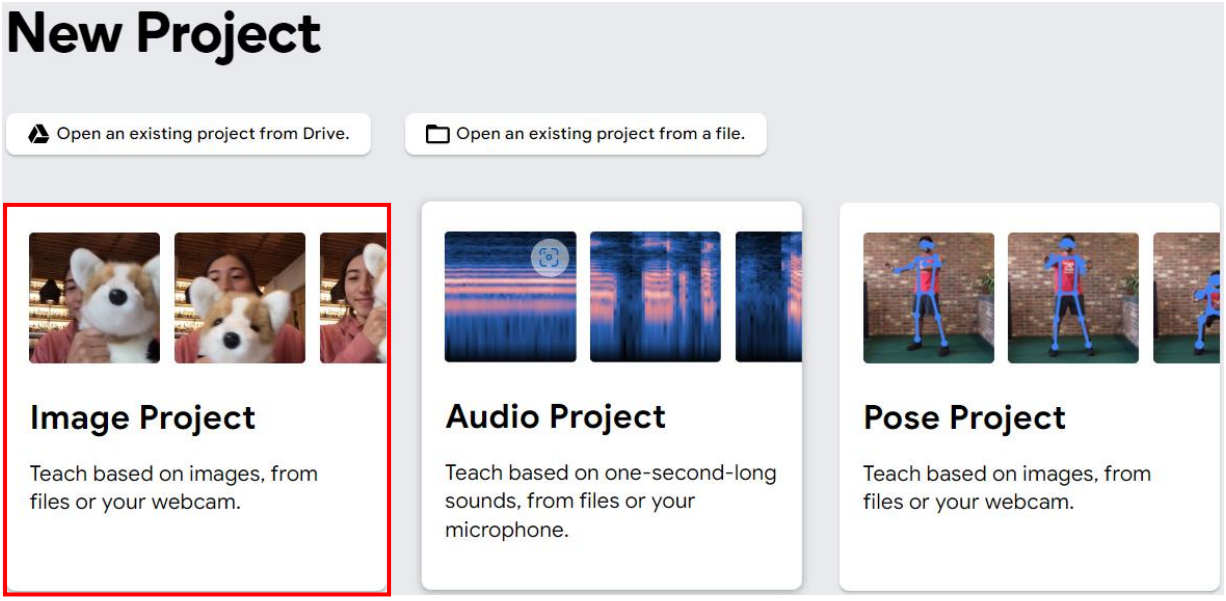
```
{
    "update_id": 834950972,
    "message": {
        "message_id": 2602690,
        "from": {
            "id": 1323657699,
            "is_bot": false,
            "first_name": "Nguy\u1ec5n V\u0103n",
            "last_name": "Tri\u1ec7u",
            "username": "hoangnam9x",
            "language_code": "vi"
        },
        "chat": {
            "id": 1323657699,
            "first_name": "Nguy\u1ec5n V\u0103n",
            "last_name": "Tri\u1ec7u",
            "username": "hoangnam9x",
            "type": "private"
        },
        "date": 1700295852,
        "text": "/start",
        "entities": [
            {
                "offset": 0,
                "length": 6,
                "type": "bot_command"
            }
        ]
    }
}
```


4. <u>TRAINING MODEL TEACHABLE MACHINE</u>:

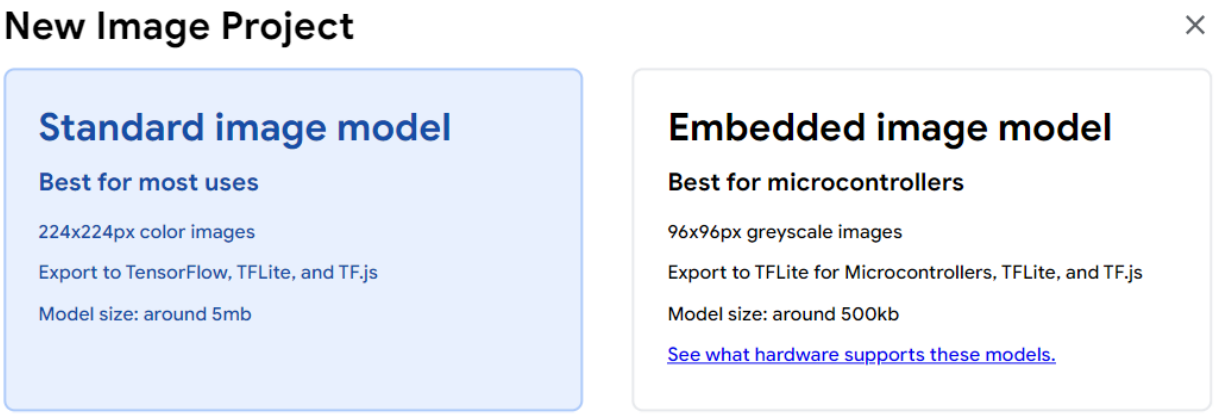
– **<u>BƯỚC 1</u>:** Vào trang chủ của Teachabke Machine theo đường dẫn: https://teachablemachine.withgoogle.com/, ấn **"GET STARTED"**.



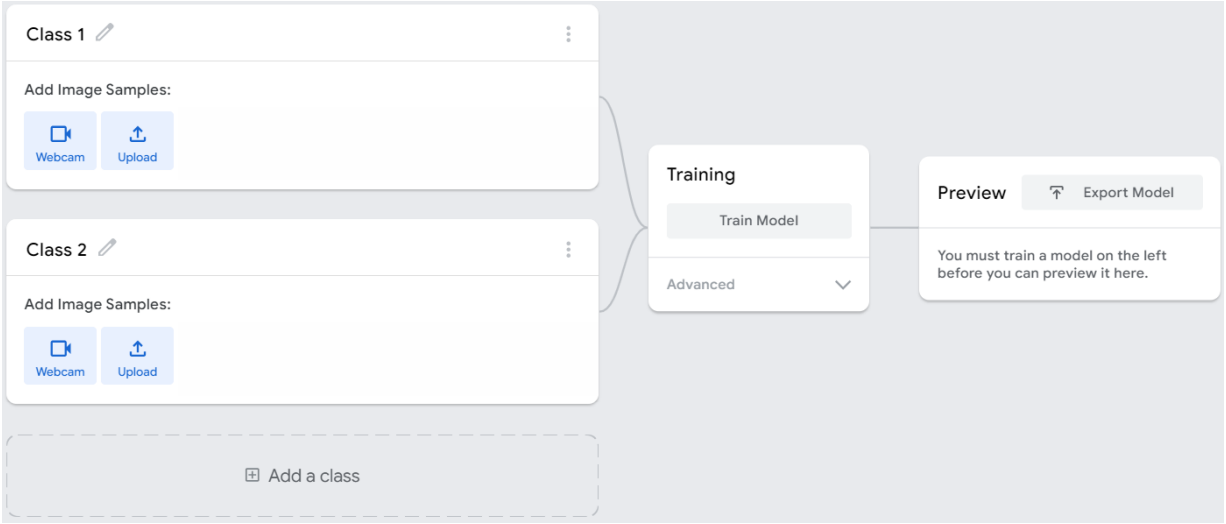
– **<u>BƯỚC 2</u>:** Chọn **"Image Project"** để tạo một project training dựa trên hình ảnh, sau đó chọn **"Standard image model"**:



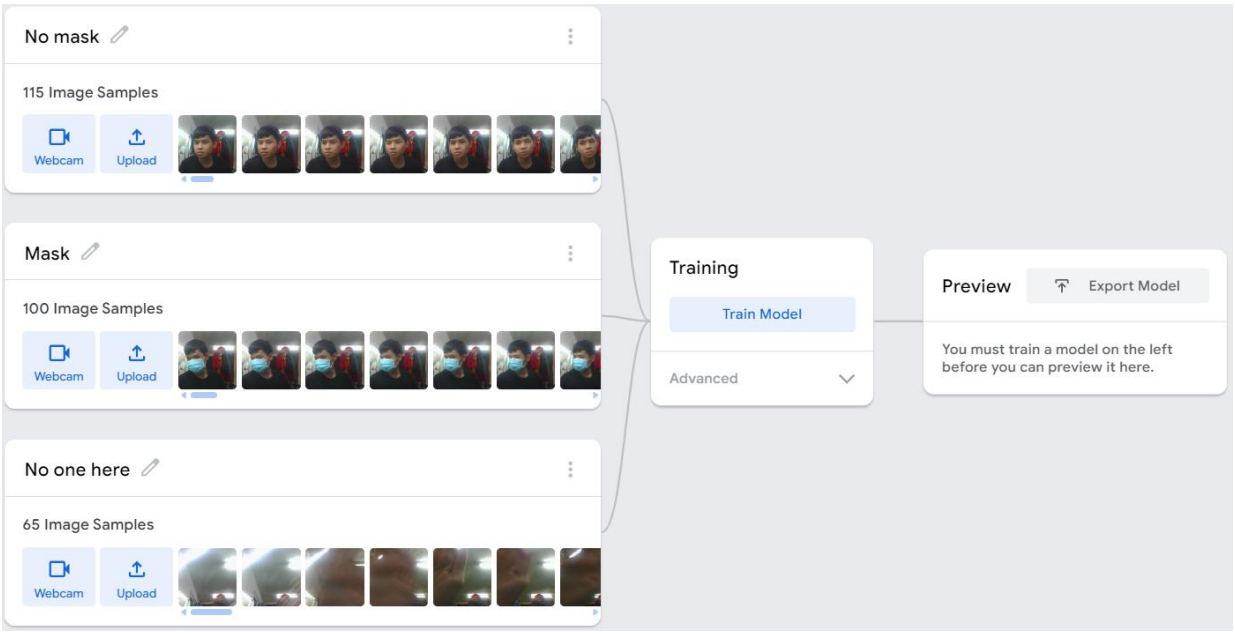
– **<u>BƯỚC 3</u>:** Chọn **"Standard image model"**.



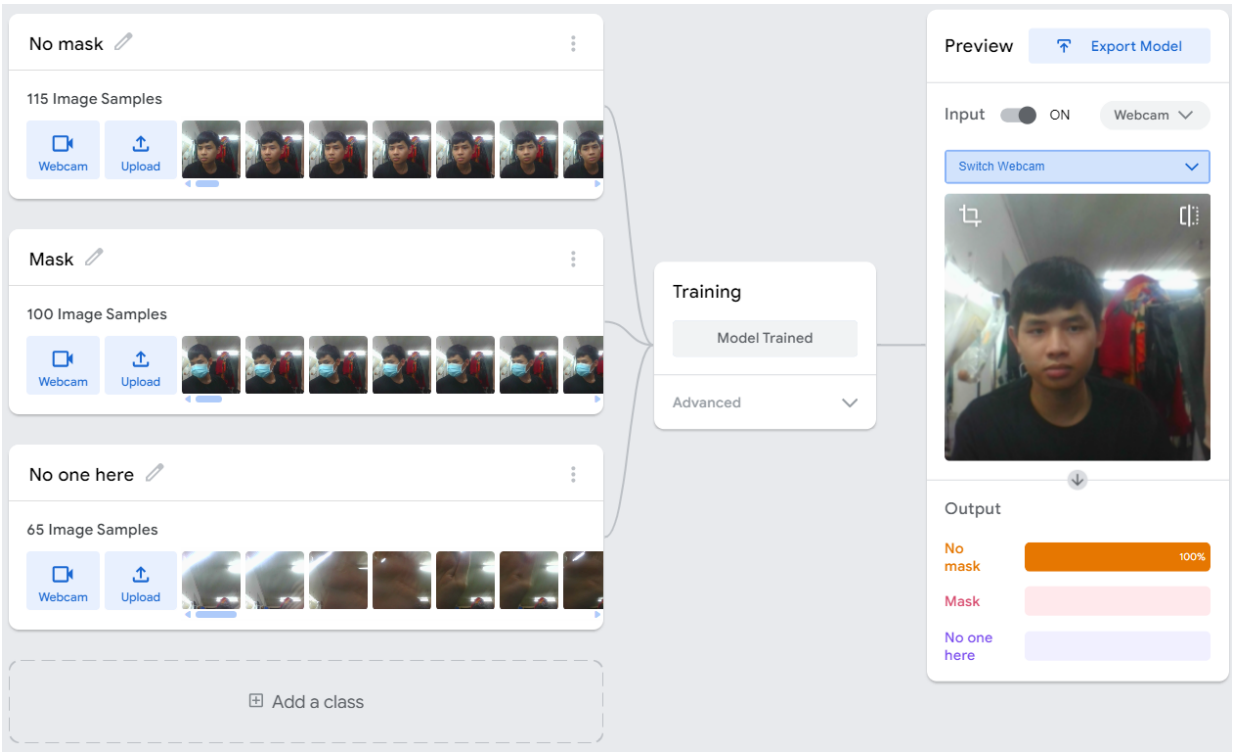
– **<u>BƯỚC 4</u>:** Giao diện chính xuất hiện. Sau đó ta hiến hành tạo ra các lớp và tải lên những hình ảnh cùng loại với lớp đó để tiến hành training model.



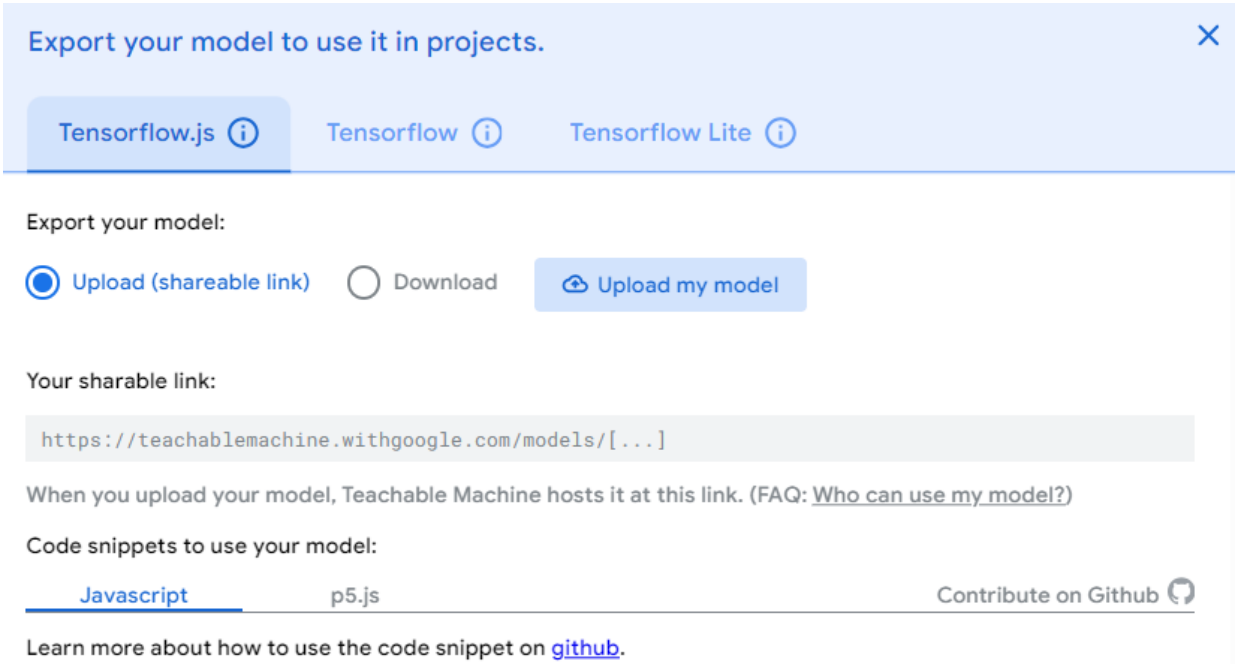
Tiếp theo, ta chọn “Train Model” để bắt đầu quá trình huấn luyện mô hình nhận diện đeo khẩu trang.



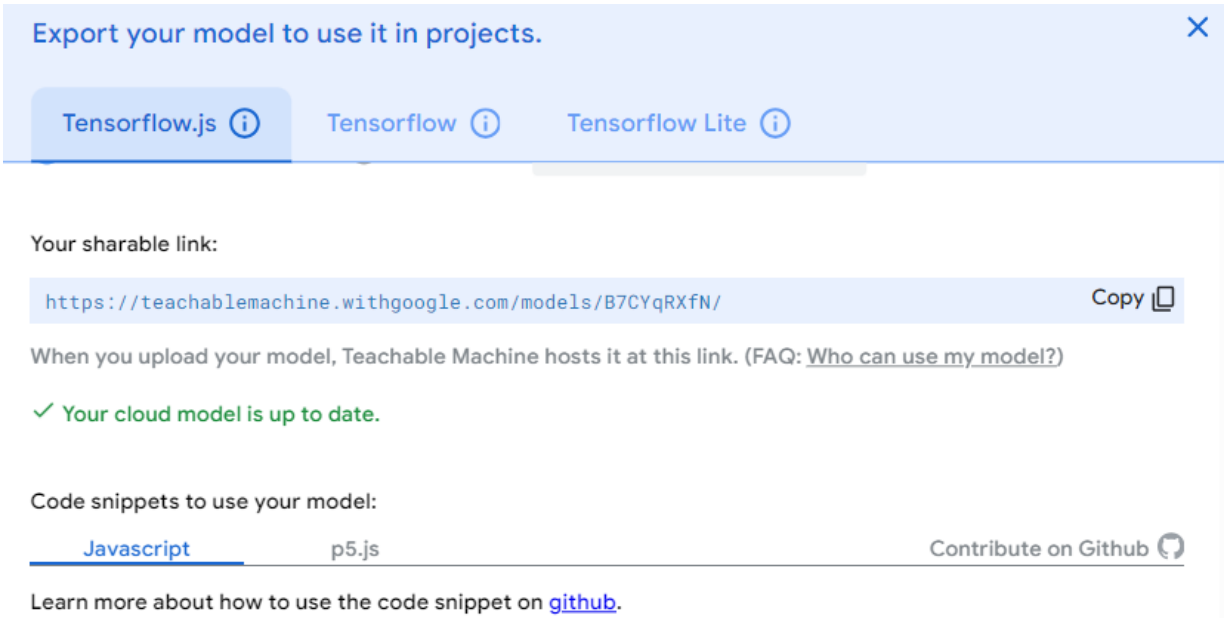
Tiếp, ta chọn “Export Model”.



Cuối cùng, ta chọn **"Upload my model"** để tải mô hình lên.



Sau đó ta có một link: https://teachablemachine.withgoogle.com/models/B7CYqRXfN/, đó chính là liên kết dẫn đến mô hình mà chúng ta đã training.



5. CODE:

a. facemask.ino:

– **BƯỚC 1:** Tạo một File mới trong Arduino IDE: **facemask.ino**

– **BƯỚC 2:** Khai báo biến **ssid** và **password** để lưu thông tin mạng Wi–Fi mà ESP32–CAM sẽ kết nối.

Khai báo biến **apssid** và **appassword** để lưu thông tin về điểm truy cập mà ESP32–CAM sẽ tạo.

```
const char* ssid       = "Dien Dep Trai Nhat KTX";
const char* password   = "hoithangthanh";
const char* apssid     = "esp32-cam";
const char* appassword = "12345678";
```

– **BƯỚC 3:** Khai báo các thư viện cần thiết.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "esp_camera.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "UniversalTelegramBot.h"
#include "ArduinoJson.h"
#include <Arduino.h>
#include "index.h"
```

– **BƯỚC 4:** Định nghĩa một số biến và khởi tạo một đối tượng bot Telegram.

BOTtoken: giá trị mã token của BotFather vừa tạo ở trên.

CHAT_ID: giá trị ID của chat vừa lấy.

flashState: trạng thái của đèn flash của camera ESP32–CAM. (tắt đèn)

botRequestDelay: độ trễ giữa việc gửi các yêu cầu đến bot Telegram. Giúp ngăn chặn việc gửi quá nhiều yêu cầu đến bot Telegram, khiến bot bị quá tải.

sendPhoto: biến kiểm tra đã đến lúc gửi ảnh mới tới tài khoản Telegram hay chưa. Mặc định biến được đặt bằng **false**.

WiFiClientSecure: một đối tượng được sử dụng cho giao tiếp HTTPS bảo mật giữa ESP32 và API Telegram. HTTPS là một giao thức bảo mật mã hóa dữ liệu được truyền giữa ESP32 và API Telegram, ngăn chặn việc bị đánh cắp bởi bên thứ ba.

UniversalTelegramBot bot(BOTtoken, clientTCP); tạo đối tượng bot với **BOTToken** và **clientTCP** đã được định nghĩa.

```
String BOTtoken = "6714097534:AAEKX7nQaIKS0ncID4_ahN4V9Pi16ANKVrs";
String CHAT_ID = "1323657699";
bool flashState = LOW;
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

bool sendPhoto = false;
WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);
```

– **BƯỚC 5:** Định nghĩa các chân GPIO trên thiết bị ESP32–CAM.

```
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#define FLASH_LED_PIN      4
```

– **BƯỚC 6:** Tạo một đối tượng **WiFiServer** có tên **server** và gán cho nó cổng 80.

```
WiFiServer server(80);
```

– **BƯỚC 7:** Khởi tạo ESP32–CAM:

WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); vô hiệu hóa bảo vệ chống cúp điện của ESP32 để tránh việc thiết bị bị khởi động lại khi có cúp điện.

Serial.begin(115200); khởi tạo giao diện Serial, tốc độ truyền 115200 bps.

Serial.setDebugOutput(true); bật chế độ debug output để hiển thị thông báo debug trên Serial Monitor.

camera_config_t config; tạo một biến cấu hình camera config.

psramFound(): kiểm tra xem có RAM psram hay không.

Nếu có psram, đặt kích thước khung hình là UXGA và chất lượng JPEG là 10.

Nếu không, đặt kích thước khung hình là SVGA và chất lượng JPEG là 12.

esp_err_t err = esp_camera_init(&config); khởi tạo camera với cấu hình đã xác định. Nếu có lỗi, in ra thông báo và khởi động lại thiết bị.

```
void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  if(psramFound()) {
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;  //0-63 lower number means higher quality
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;  //0-63 lower number means higher quality
    config.fb_count = 1;
  }

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    delay(1000);
    ESP.restart();
  }
```

sensor_t * s = esp_camera_sensor_get(); sử dụng con trỏ s để lấy thông tin về camera.

s–>set_framesize(s, FRAMESIZE_QVGA); đặt kích thước khung hình thành QVGA (320x240 pixel).

ledcAttachPin(4, 4); thiết lập kết nối giữa chân GPIO 4 và kênh LEDC 4.

ledcSetup(4, 5000, 8); cấu hình kênh LEDC 4 với tần số 5000 Hz và độ phân giải 8 bit. (điều khiển độ sáng của LED)

```
sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_QVGA);

ledcAttachPin(4, 4);
ledcSetup(4, 5000, 8);
```

WiFi.mode(WIFI_AP_STA); thiết lập chế độ hoạt động của thiết bị Wi–Fi là Access Point (AP) và Station (STA) đồng thời.

WiFi.begin(ssid, password); bắt đầu quá trình kết nối đến mạng Wi–Fi đã thiết lập từ các bước khai báo đầu tiên ở trên.

clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT); thiết lập chứng chỉ gốc của Telegram để xác thực máy chủ Telegram khi gửi tin nhắn.

```
WiFi.mode(WIFI_AP_STA);
WiFi.begin(ssid, password);
clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);
delay(1000);
Serial.println("");
Serial.print("Connecting to ");
Serial.println(ssid);
```

while (WiFi.status() != WL_CONNECTED): lặp lại cho đến khi thiết bị kết nối thành công.

```
long int StartTime = millis();
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    if ((StartTime+10000) < millis()) break;
}
```

if (WiFi.status() == WL_CONNECTED): kiểm tra xem kết nối đã thành công hay không.

Nếu kết nối thành công, thiết lập một Access Point với tên mạng là địa chỉ IP. Vòng **for:** chớp đèn LED 5 lần để thông báo kết nối thành công.

Nếu không thành công, thiết lập một Access Point với tên mạng là địa chỉ IP của thiết bị AP. Vòng **for**: chớp đèn LED 2 lần để thông báo kết nối thất bại.

```
if (WiFi.status() == WL_CONNECTED) {
  WiFi.softAP((WiFi.localIP().toString() + "_" + (String)apssid).c_str(), appassword);
  Serial.println("");
  Serial.println("STAIP address: ");
  Serial.println(WiFi.localIP());

  for (int i = 0; i < 5; i++) {
    ledcWrite(4, 10);
    delay(200);
    ledcWrite(4, 0);
    delay(200);
  }
}
else {
  WiFi.softAP((WiFi.softAPIP().toString() + "_" + (String)apssid).c_str(), appassword);
    Serial.println("Connect wifi fail");
  for (int i = 0; i < 2; i++) {
    ledcWrite(4, 10);
    delay(1000);
    ledcWrite(4, 0);
    delay(1000);
  }
}
```

– **BƯỚC 8:** Xử lý các tin nhắn mới đến:

Serial.println(numNewMessages); in ra số lượng tin nhắn mới cần xử lý.

Vòng **for**: duyệt qua từng tin nhắn mới để xử lý.

+ **String chat_id = String(bot.messages[i].chat_id);** lấy ID cuộc trò chuyện của tin nhắn hiện tại và chuyển nó thành một đối tượng String.
+ **if (chat_id != CHAT_ID):** kiểm tra xem ID cuộc trò chuyện của tin nhắn hiện tại có khớp với hằng số CHAT_ID hay không. Nếu ID không khớp, báo lỗi.
+ **String text = bot.messages[i].text;** lấy nội dung của tin nhắn.
+ **String from_name = bot.messages[i].from_name;** lấy tên người gửi.
+ **if (text == "/start"):** nếu nội dung của tin nhắn là **"/start"**, tạo ra thông báo chào mừng và hướng dẫn sử dụng.
+ **if (text == "/flash"):** nếu nội dung của tin nhắn là **"/flash"** , bật tắt trạng thái đèn LED flash và in ra màn hình console thông báo.
+ **if (text == "/photo"):** nếu nội dung của tin nhắn là **"/photo"** , đặt biến **sendPhoto** thành **true**.

```
void handleNewMessages(int numNewMessages) {
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);
  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }
    String text = bot.messages[i].text;
    Serial.println(text);
    String from_name = bot.messages[i].from_name;
    if (text == "/start") {
      String welcome = "Welcome , " + from_name + "\n";
      welcome += "Use the following commands to interact with the ESP32-CAM \n";
      welcome += "/photo : takes a new photo\n";
      welcome += "/flash : toggles flash LED \n";
      bot.sendMessage(CHAT_ID, welcome, "");
    }
    if (text == "/flash") {
      flashState = !flashState;
      digitalWrite(FLASH_LED_PIN, flashState);
      Serial.println("Change flash LED state");
    }
    if (text == "/photo") {
      sendPhoto = true;
      Serial.println("New photo request");
    }
  }
}
```

– **<u>BƯỚC 9</u>:** Chụp ảnh từ ESP32–CAM và gửi ảnh lên Telegram thông qua API.

const char* myDomain = "api.telegram.org"; định nghĩa tên miền của Telegram API, tức địa chỉ domain của dịch vụ Telegram API.

String getAll = "";

String getBody = "";

lưu trữ dữ liệu nhận được từ server Telegram.

camera_fb_t * fb = NULL;

fb = esp_camera_fb_get();

esp_camera_fb_return(fb);

fb = NULL;

fb = esp_camera_fb_get();

chụp ảnh từ camera sử dụng ESP32–CAM. Nếu quá trình này thành công, ảnh được lưu trữ trong biến **fb**.

if(!fb): nếu chụp ảnh không thành công, in ra màn hình Serial thông báo lỗi và khởi động lại ESP32–CAM.

```
String sendPhotoTelegram() {
  const char* myDomain = "api.telegram.org";
  String getAll  = "";
  String getBody = "";
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  esp_camera_fb_return(fb); // dispose the buffered image
  fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }
```

Serial.println("Connect to " + String(myDomain)); đang kết nối tới API Telegram.

if (clientTCP.connect(myDomain, 443)): kết nối đến server của Telegram API thông qua giao thức HTTPS.

+ **String head:** định nghĩa một chuỗi chứa tiêu đề của yêu cầu HTTP. Tiêu đề này bao gồm ID cuộc trò chuyện và tên của tệp ảnh.
+ **String tail:** định nghĩa một chuỗi chứa phần còn lại của yêu cầu HTTP. Phần này bao gồm dữ liệu ảnh.
+ **size_t imageLen = fb –> len;** lấy độ dài của dữ liệu ảnh.
+ **size_t extraLen = head.length() + tail.length();** lấy độ dài của tiêu đề và phần còn lại của yêu cầu HTTP.
+ **size_t totalLen = imageLen + extraLen;** tính độ dài của yêu cầu HTTP.
+ **clientTCP.println("POST /bot" + BOTtoken + "/sendPhoto HTTP/1.1");**

clientTCP.println("Host: " + String(myDomain));

clientTCP.println("Content–Length: " + String(totalLen));

clientTCP.println("Content–Type: multipart/form–data; boundary = RandomNerdTutorials");

clientTCP.print(head);

gửi các dòng yêu cầu HTTP POST lên server Telegram, bao gồm cả header và thông tin về độ dài của dữ liệu.

```
Serial.println("Connect to " + String(myDomain));
if (clientTCP.connect(myDomain, 443)) {
  Serial.println("Connection successful");
  String head = "--RandomNerdTutorials\r\n"
            "Content-Disposition: form-data; name=\"chat_id\"; \r\n"
            "\r\n" + CHAT_ID + "\r\n"
            "--RandomNerdTutorials\r\n"
            "Content-Disposition: form-data; name=\"photo\"; filename=\"esp32-cam.jpg\"\r\n"
            "Content-Type: image/jpeg\r\n"
            "\r\n";
  String tail = "\r\n--RandomNerdTutorials--\r\n";
  size_t imageLen = fb -> len;
  size_t extraLen = head.length() + tail.length();
  size_t totalLen = imageLen + extraLen;
  clientTCP.println("POST /bot" + BOTtoken + "/sendPhoto HTTP/1.1");
  clientTCP.println("Host: " + String(myDomain));
  clientTCP.println("Content-Length: " + String(totalLen));
  clientTCP.println("Content-Type: multipart/form-data; boundary=RandomNerdTutorials");
  clientTCP.println();
  clientTCP.print(head);
```

+ **uint8_t *fbBuf = fb –> buf;** lấy con trỏ đến bộ đệm ảnh.

  size_t fbLen = fb –> len; lấy độ dài của bộ đệm ảnh.

+ **for (size_t n = 0; n < fbLen; n = n + 1024):** gửi dữ liệu ảnh từ biến **fb** lên server Telegram dưới dạng dữ liệu nhị phân (binary data). Dữ liệu được gửi theo các phần nhỏ (chunks) có kích thước là 1024 bytes.
+ **clientTCP.print(tail);** gửi phần tail của request HTTP POST.
+ **esp_camera_fb_return(fb);** giải phóng bộ nhớ đã sử dụng cho ảnh.
+ **int waitTime = 10000;** thời gian cho việc chờ phản hồi từ API Telegram.
+ **long startTimer = millis();** lấy thời gian hiện tại theo mili giây.
+ **boolean state = false;** theo dõi các tiêu đề phản hồi đã được đọc chưa.

```
uint8_t *fbBuf = fb -> buf;
size_t fbLen = fb -> len;
for (size_t n = 0; n < fbLen; n = n + 1024) {
  if (n + 1024 < fbLen) {
    clientTCP.write(fbBuf, 1024);
    fbBuf += 1024;
  }
  else if (fbLen % 1024 > 0) {
    size_t remainder = fbLen % 1024;
    clientTCP.write(fbBuf, remainder);
  }
}
clientTCP.print(tail);
esp_camera_fb_return(fb);
int waitTime = 10000;   // timeout 10 seconds
long startTimer = millis();
boolean state = false;
```

+ **while ((startTimer + waitTime) > millis()):** xử lý phản hồi từ API Telegram.

```
    while ((startTimer + waitTime) > millis()){
      Serial.print(".");
      delay(100);
      while (clientTCP.available()) {
        char c = clientTCP.read();
        if (state == true) getBody += String(c);
        if (c == '\n') {
          if (getAll.length()==0) state = true;
          getAll = "";
        }
        else if (c != '\r')
          getAll += String(c);
        startTimer = millis();
      }
      if (getBody.length()>0) break;
    }
    clientTCP.stop();
    Serial.println(getBody);
  }
  else {
    getBody = "Connected to api.telegram.org failed.";
    Serial.println("Connected to api.telegram.org failed.");
  }
  return getBody;
}
```

b. index.h:

```
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<head>
    <title>ESP32-CAMERA Teachable Machine</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link rel="stylesheet" href="https:\/\/cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
        integrity="sha384-Gn5384xqQlaoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
        crossorigin="anonymous">
    <style>
        body {
            font-family: Arial, Helvetica, sans-serif;
            background: #181818;
            color: #EFEFEF;
            font-size: 16px;
            display: none;
        }
        h2 {
            font-size: 18px
        }
        section.main {
            display: flex
        }
        #menu,
        section.main {
            flex-direction: column
        }
        #menu {
            display: flex;
            flex-wrap: nowrap;
            min-width: 450px;
            background: #363636;
            padding: 8px;
            border-radius: 4px;
            margin-top: -10px;
            margin-right: 10px
        }
        #content {
            display: flex;
            flex-wrap: wrap;
            align-items: stretch
        }
        figure {
            padding: 0;
            margin: 0;
            -webkit-margin-before: 0;
            margin-block-start: 0;
            -webkit-margin-after: 0;
            margin-block-end: 0;
            -webkit-margin-start: 0;
            margin-inline-start: 0;
            -webkit-margin-end: 0;
            margin-inline-end: 0
        }
        figure img {
            display: block;
            width: 100%;
            height: auto;
            border-radius: 4px;
            margin-top: 8px
        }
```

```
@media (min-width: 800px) and (orientation:landscape) {
    #content {
        display: flex;
        flex-wrap: nowrap;
        align-items: stretch
    }
    figure img {
        display: block;
        max-width: 100%;
        max-height: calc(100vh - 40px);
        width: auto;
        height: auto
    }
    figure {
        padding: 0;
        margin: 0;
        -webkit-margin-before: 0;
        margin-block-start: 0;
        -webkit-margin-after: 0;
        margin-block-end: 0;
        -webkit-margin-start: 0;
        margin-inline-start: 0;
        -webkit-margin-end: 0;
        margin-inline-end: 0
    }
}
section#buttons {
    display: flex;
    flex-wrap: nowrap;
    justify-content: space-between
}
#nav-toggle {
    cursor: pointer;
    display: block
}
#nav-toggle-cb {
    outline: 0;
    opacity: 0;
    width: 0;
    height: 0
}
#nav-toggle-cb:checked+#menu {
    display: none
}
.input-group {
    display: flex;
    flex-wrap: nowrap;
    line-height: 22px;
    margin: 5px 0
}
.input-group>label {
    display: inline-block;
    padding-right: 10px;
    min-width: 20%
}

.input-group input,
.input-group select {
    flex-grow: 1
}
.range-max,
.range-min {
    display: inline-block;
    padding: 0 5px
}
```

```
        button {
            display: block;
            margin: 5px;
            padding: 0 12px;
            border: 0;
            line-height: 28px;
            cursor: pointer;
            color: #fff;
            background: #4135a8;
            border-radius: 5px;
            font-size: 16px;
            outline: 0
        }
        button:hover {
            background: #776ff1
        }
        button:active {
            background: #1100ff
        }
        button.disabled {
            cursor: default;
            background: #a0a0a0
        }
        input[type=range] {
            -webkit-appearance: none;
            width: 100%;
            height: 22px;
            background: #363636;
            cursor: pointer;
            margin: 0
        }
        input[type=range]:focus {
            outline: 0
        }
        input[type=range]::-webkit-slider-runnable-track {
            width: 100%;
            height: 2px;
            cursor: pointer;
            background: #EFEFEF;
            border-radius: 0;
            border: 0 solid #EFEFEF
        }
        input[type=range]::-webkit-slider-thumb {
            border: 1px solid rgba(0, 0, 30, 0);
            height: 22px;
            width: 22px;
            border-radius: 50px;
            background: #2c1497;
            cursor: pointer;
            -webkit-appearance: none;
            margin-top: -11.5px
        }
        input[type=range]:focus::-webkit-slider-runnable-track {
            background: #EFEFEF
        }
        input[type=range]::-moz-range-track {
            width: 100%;
            height: 2px;
            cursor: pointer;
            background: #EFEFEF;
            border-radius: 0;
            border: 0 solid #EFEFEF
        }
```

```
        input[type=range]::-moz-range-thumb {
            border: 1px solid rgba(0, 0, 30, 0);
            height: 22px;
            width: 22px;
            border-radius: 50px;
            background: #2c1497;
            cursor: pointer
        }
        input[type=range]::-ms-track {
            width: 100%;
            height: 2px;
            cursor: pointer;
            background: 0 0;
            border-color: transparent;
            color: transparent
        }
        input[type=range]::-ms-fill-lower {
            background: #EFEFEF;
            border: 0 solid #EFEFEF;
            border-radius: 0
        }
        input[type=range]::-ms-fill-upper {
            background: #EFEFEF;
            border: 0 solid #EFEFEF;
            border-radius: 0
        }
        input[type=range]::-ms-thumb {
            border: 1px solid rgba(0, 0, 30, 0);
            height: 22px;
            width: 22px;
            border-radius: 50px;
            background: #2c1497;
            cursor: pointer;
            height: 2px
        }
        input[type=range]:focus::-ms-fill-lower {
            background: #EFEFEF
        }
        input[type=range]:focus::-ms-fill-upper {
            background: #363636
        }
        .switch {
            display: block;
            position: relative;
            line-height: 22px;
            font-size: 16px;
            height: 22px
        }
        .switch input {
            outline: 0;
            opacity: 0;
            width: 0;
            height: 0
        }
        .slider {
            width: 50px;
            height: 22px;
            border-radius: 22px;
            cursor: pointer;
            background-color: grey
        }
        .slider,
        .slider:before {
            display: inline-block;
            transition: .4s
        }
```

```
        .slider:before {
            position: relative;
            content: "";
            border-radius: 50%;
            height: 16px;
            width: 16px;
            left: 4px;
            top: 3px;
            background-color: #fff
        }
        input:checked+.slider {
            background-color: #2c1497
        }
        input:checked+.slider:before {
            -webkit-transform: translateX(26px);
            transform: translateX(26px)
        }
        select {
            border: 1px solid #363636;
            font-size: 14px;
            height: 22px;
            outline: 0;
            border-radius: 5px
        }
        .image-container {
            position: relative;
            min-width: 160px
        }
        .close {
            position: absolute;
            right: 5px;
            top: 5px;
            background: #2c1497;
            width: 16px;
            height: 16px;
            border-radius: 100px;
            color: #fff;
            text-align: center;
            line-height: 18px;
            cursor: pointer
        }
        .hidden {
            display: none
        }
    </style>
    <script src="https:\/\/ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
    <script src="https:\/\/cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"></script>
    <script
        src="https:\/\/cdn.jsdelivr.net/npm/@teachablemachine/image@0.8/dist/teachablemachine-image.min.js"></script>
    <script src="https:\/\/cdn.jsdelivr.net/npm/@teachablemachine/pose@0.8/dist/teachablemachine-pose.min.js"></script>
</head>

<body>
    <div id="container">

        <div class="row" style="margin-top: 50px;">
            <div class="col-2">
            </div>
            <div class="col-5">

                <div id="logo">
                    <label for="nav-toggle-cb" id="nav-toggle">&#9776;  Menu</label>
                </div>
                <div id="content">
                    <div id="sidebar">
                        <input type="checkbox" id="nav-toggle-cb">
                        <nav id="menu">
```

```
                        <div class="input-group">
                            <label for="kind">Model kind:</label>
                            <select id="kind">
                                <option value="image">image</option>
                                <option value="pose">pose</option>
                            </select>
                        </div>
                        <div class="input-group" style="display:none;">
                            <label for="modelPath">Model path:</label>
                            <input type="text" id="modelPath"
                                value="https://teachablemachine.withgoogle.com/models/B7CYqRXfN">
                        </div>
                        <div class="input-group" style="display:none;">
                            <label for="btnModel"></label>
                            <button type="button" id="btnModel" onclick="LoadModel();">Start Recognition</button>
                        </div>
                        <div class="input-group" id="mirrorimage-group">
                            <label for="mirrorimage">Resolution</label>
                            <select id="mirrorimage" class="default-action">
                                <option value="1">yes</option>
                                <option value="0">no</option>
                            </select>
                        </div>
                        <div class="input-group" id="flash-group">
                            <label for="flash">Flash</label>
                            <div class="range-min">0</div>
                            <input type="range" id="flash" min="0" max="255" value="0" class="default-action">
                            <div class="range-max">255</div>
                        </div>
                        <div class="input-group" id="framesize-group">
                            <label for="framesize">Resolution</label>
                            <select id="framesize" class="default-action">
                                <option value="10">UXGA(1600x1200)</option>
                                <option value="9">SXGA(1280x1024)</option>
                                <option value="8">XGA(1024x768)</option>
                                <option value="7">SVGA(800x600)</option>
                                <option value="6">VGA(640x480)</option>
                                <option value="5" selected="selected">CIF(400x296)</option>
                                <option value="4">QVGA(320x240)</option>
                                <option value="3">HQVGA(240x176)</option>
                                <option value="0">QQVGA(160x120)</option>
                            </select>
                        </div>
                        <div class="input-group" id="quality-group">
                            <label for="quality">Quality</label>
                            <div class="range-min">10</div>
                            <input type="range" id="quality" min="10" max="63" value="10" class="default-action">
                            <div class="range-max">63</div>
                        </div>
                        <div class="input-group" id="brightness-group">
                            <label for="brightness">Brightness</label>
                            <div class="range-min">-2</div>
                            <input type="range" id="brightness" min="-2" max="2" value="0" class="default-action">
                            <div class="range-max">2</div>
                        </div>
                        <div class="input-group" id="contrast-group">
                            <label for="contrast">Contrast</label>
                            <div class="range-min">-2</div>
                            <input type="range" id="contrast" min="-2" max="2" value="0" class="default-action">
                            <div class="range-max">2</div>
                        </div>
                        <div class="input-group" id="rotate-group">
                            <label for="rotate">Rotate:</label>
                            <select
                                onchange="document.getElementById('canvas').style.transform='rotate('+this.value+')';">
                                <option value="0deg">0deg</option>
                                <option value="90deg">90deg</option>
                                <option value="180deg">180deg</option>
                                <option value="270deg">270deg</option>
                            </select>
                        </div>
                    </nav>
                </div>
            </div>
```

```
                <div id="result" style="color:red">
                    <div>
                    </div>
                </div>
            </div>
            <div class="col-5" style="margin-top: 50px;">
                <img id="ShowImage" src="" style="display:none">
                <canvas id="canvas" style="display: none;"></canvas>

                <section id="buttons">
                    <table>
                        <tr>
                            <td><button type="button" id="restart">Restart</button></td>
                            <td colspan="2"><button type="button" id="getStill" style="display:none">Get Still</button>
                            </td>
                        </tr>
                    </table>
                </section>
            </div>
</body>

</html>

<script>
    var getStill = document.getElementById('getStill');
    var ShowImage = document.getElementById('ShowImage');
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    var mirrorimage = document.getElementById("mirrorimage");
    var flash = document.getElementById('flash');
    var modelPath = document.getElementById('modelPath');
    var result = document.getElementById('result');
    var kind = document.getElementById('kind');
    var myTimer;
    var restartCount = 0;

    async function LoadModel() {
        if (modelPath.value == "") {
            result.innerHTML = "Please input model path.";
            return;
        }

        result.innerHTML = "Please wait for loading model.";

        const URL = modelPath.value;
        const modelURL = URL + "/model.json";
        const metadataURL = URL + "/metadata.json";

        if (kind.value == "image") {
            Model = await tmImage.load(modelURL, metadataURL);
        }
        else if (kind.value == "pose") {
            Model = await tmPose.load(modelURL, metadataURL);
        }
        maxPredictions = Model.getTotalClasses();
        result.innerHTML = "";

        getStill.style.display = "block";
        getStill.click();
        document.body.style.display = "block";
    }
    document.addEventListener('DOMContentLoaded', function() {
    LoadModel();
    });
    getStill.onclick = function (event) {
        clearInterval(myTimer);
        myTimer = setInterval(function () { error_handle(); }, 5000);
        ShowImage.src = location.origin + '/?getstill=' + Math.random();
    }

```

```
    function error_handle() {
        restartCount++;
        clearInterval(myTimer);
        if (restartCount <= 2) {
            //message.innerHTML = "Get still error. <br>Restart ESP32-CAM "+restartCount+" times.";
            myTimer = setInterval(function () { getStill.click(); }, 6000);
        }
        else
            message.innerHTML = "Get still error. <br>Please close the page and check ESP32-CAM.";
    }

    ShowImage.onload = function (event) {
        clearInterval(myTimer);
        restartCount = 0;
        canvas.setAttribute("width", ShowImage.width);
        canvas.setAttribute("height", ShowImage.height);
        canvas.style.display = "block";

        if (mirrorimage.value == 1) {
            context.translate((canvas.width + ShowImage.width) / 2, 0);
            context.scale(-1, 1);
            context.drawImage(ShowImage, 0, 0, ShowImage.width, ShowImage.height);
            context.setTransform(1, 0, 0, 1, 0, 0);
        }
        else
            context.drawImage(ShowImage, 0, 0, ShowImage.width, ShowImage.height);
        predict();
    }

    restart.onclick = function (event) {
        fetch(location.origin + '/?restart=stop');
    }
    framesize.onclick = function (event) {
        fetch(document.location.origin + '/?framesize=' + this.value + ';stop');
    }
    flash.onchange = function (event) {
        fetch(location.origin + '/?flash=' + this.value + ';stop');
    }
    quality.onclick = function (event) {
        fetch(document.location.origin + '/?quality=' + this.value + ';stop');
    }
    brightness.onclick = function (event) {
        fetch(document.location.origin + '/?brightness=' + this.value + ';stop');
    }
    contrast.onclick = function (event) {
        fetch(document.location.origin + '/?contrast=' + this.value + ';stop');
    }

    async function predict() {
        var data = "";
        var maxClassName = "";
        var maxProbability = "";

        canvas.setAttribute("width", ShowImage.width);
        canvas.setAttribute("height", ShowImage.height);
        context.drawImage(ShowImage, 0, 0, ShowImage.width, ShowImage.height);

        if (kind.value == "image")
            var prediction = await Model.predict(canvas);
        else if (kind.value == "pose") {
            var { pose, posenetOutput } = await Model.estimatePose(canvas);
            var prediction = await Model.predict(posenetOutput);
        }

```

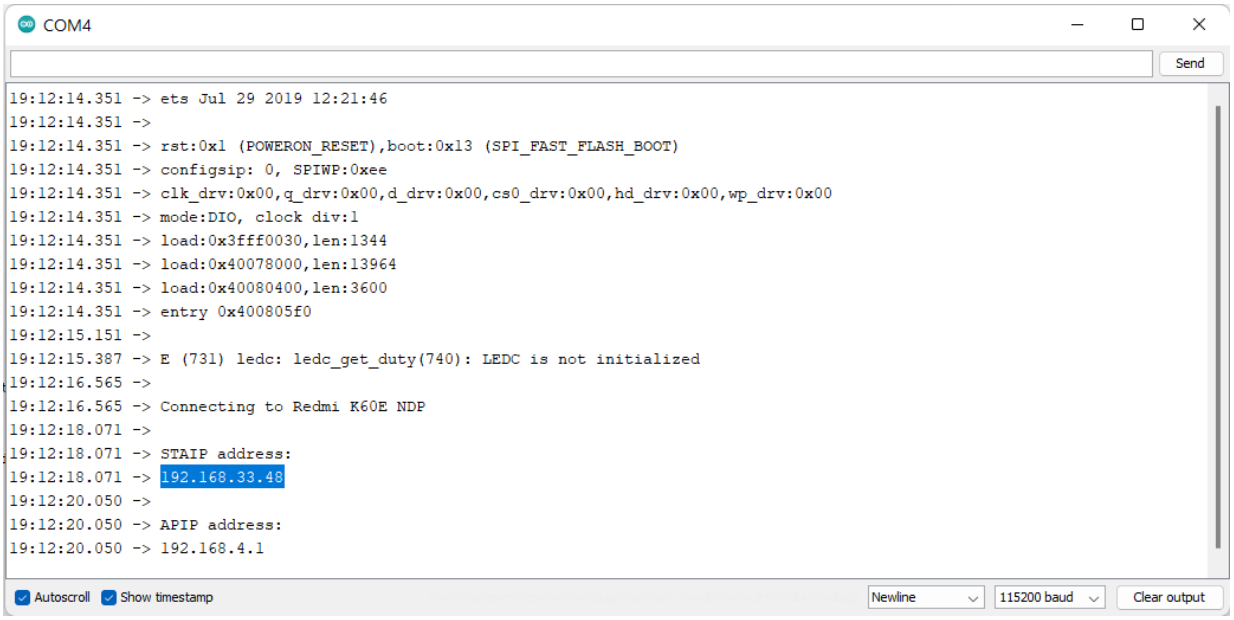
```
        if (maxPredictions > 0) {
            for (let i = 0; i < maxPredictions; i++) {
                if (i == 0) {
                    maxClassName = prediction[i].className;
                    maxProbability = prediction[i].probability;
                }
                else {
                    if (prediction[i].probability > maxProbability) {
                        maxClassName = prediction[i].className;
                        maxProbability = prediction[i].probability;
                    }
                }
                data += prediction[i].className + "," + prediction[i].probability.toFixed(2) + "<br>";
            }
            result.innerHTML = data;
            result.innerHTML += "<br>Result: " + maxClassName + "," + maxProbability;

            $.ajax({ url: document.location.origin + '/?serial=' + maxClassName + ";" + maxProbability + ';stop', async: false });
        }
        else
            result.innerHTML = "Unrecognizable";

        getStill.click();
    }
</script>
)rawliteral";
```

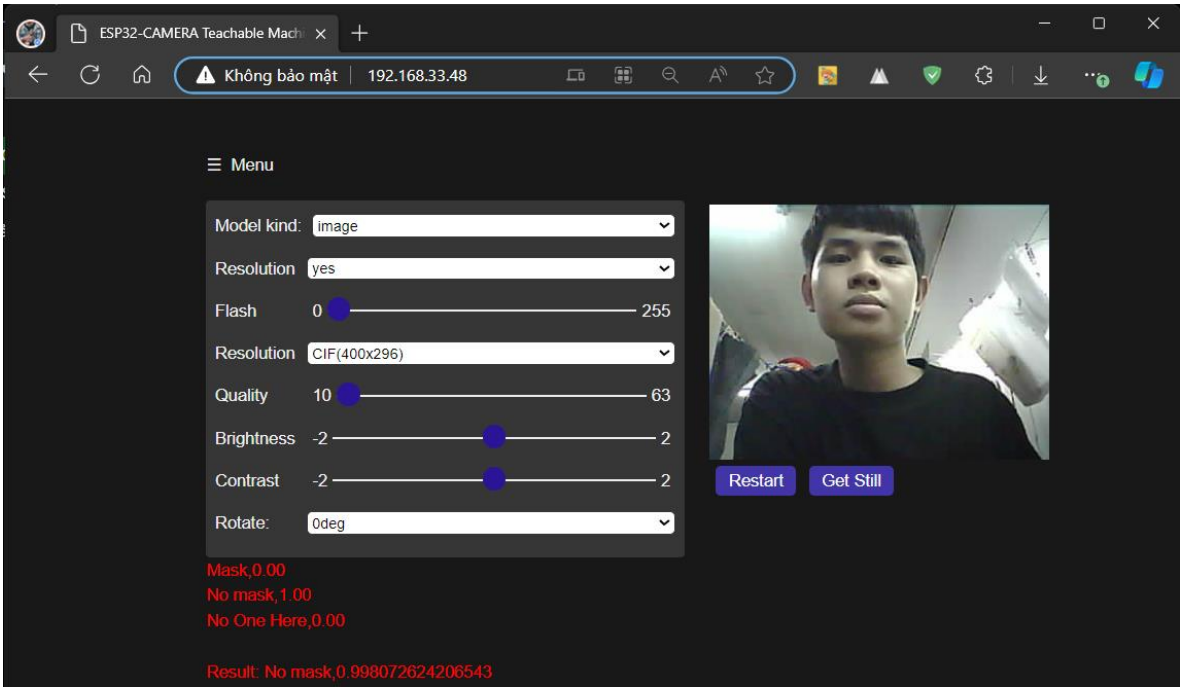
V. KẾT QUẢ:

– Sau khi upload code vào ESP32–CAM thành công, ta bật Serial Monitor trong Arduino, sau đó nhấn nút “RESET” trên mạch nạp. Truy cập địa chỉ 192.169.33.48 trong trình duyệt web.

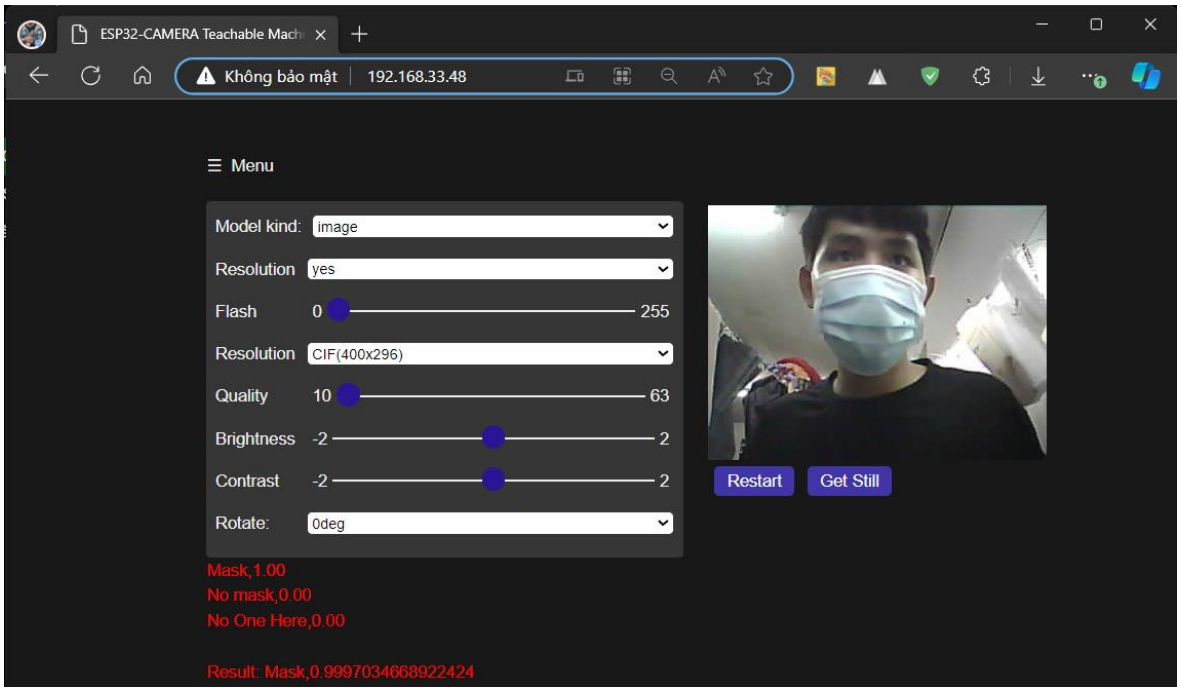


– Mô hình nhận diện được khuôn mặt người không đeo khẩu trang, có đeo khẩu trang và trường hợp không có người.

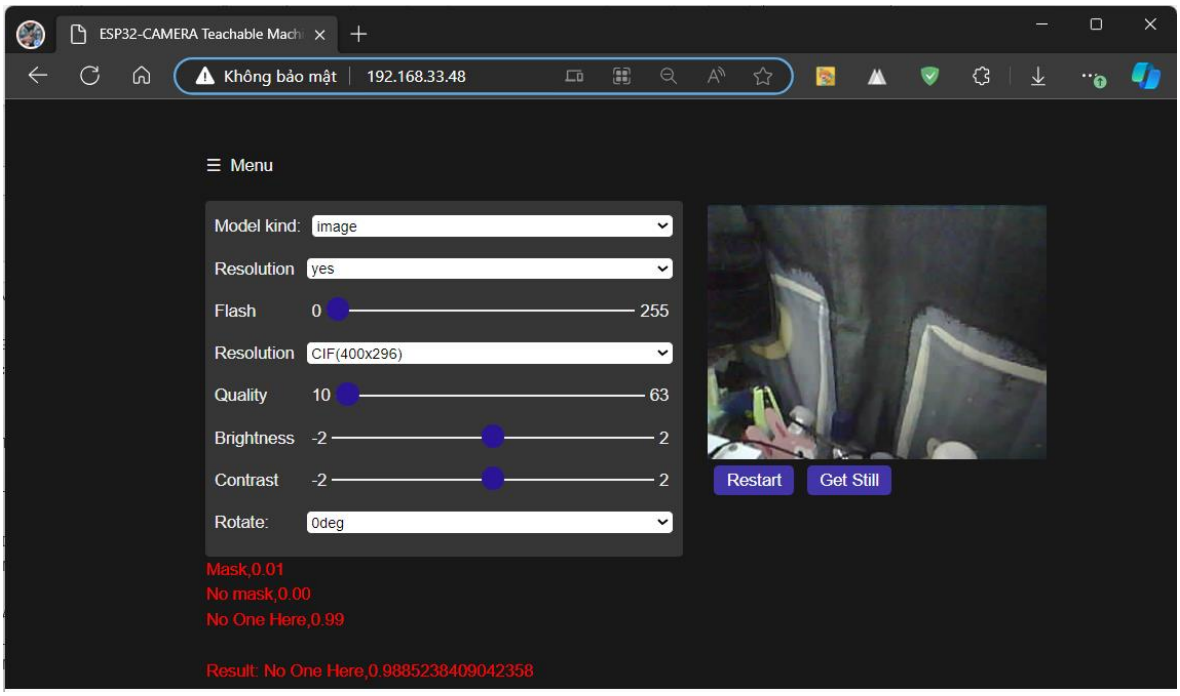
– Kết quả từ mô hình nhận diện không đeo khẩu trang là ACCURACY = 100%.



– Kết quả từ mô hình nhận diện đeo khẩu trang là ACCURACY = 100%.



– Kết quả từ mô hình nhận diện không có người là ACCURACY = 99%.



– Kết quả trả về từ BOT Telegram.